ABC INFORMATIQUE

N° 29

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Les systèmes «Expert»

L'IBM PC

Le basic Commodore

Testez vos circuits

EDITIONS ATLAS

M6062-29-12F 85FB-3,80FS-$1.95

ABC INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : (37) 35-40-23. Services administratifs et commerciaux : 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE AU NUMÉRO

Les numéros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser aux services commerciaux des ÉDITIONS ATLAS. Tél. : (37) 35-40-23.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume III, n° 29.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : Sivea.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : juillet 1984. 20847. Dépôt légal en Belgique : D/84/2783/27.

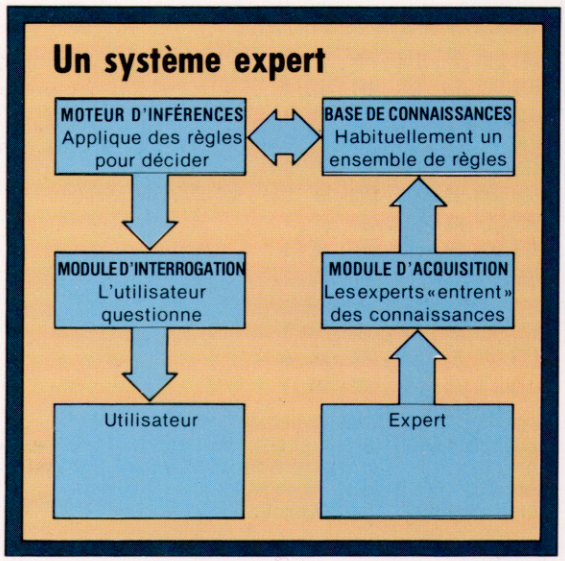
A NOS LECTEURS

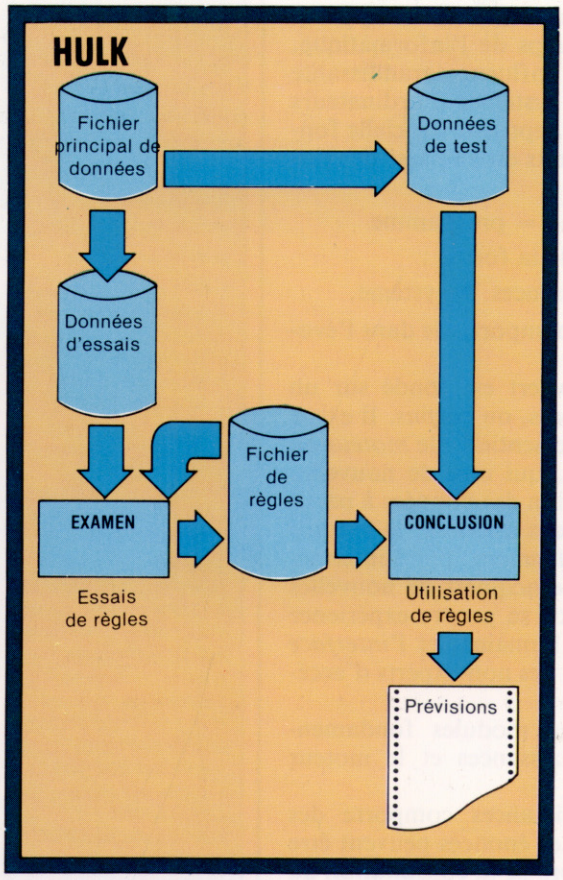
En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

L'étonnant « Hulk »

Le terme « ingénierie » connaît aujourd'hui un changement de sens : au XIXe siècle un ingénieur était quelqu'un comme Gustave Eiffel. Maintenant, « ingénierie » s'applique aussi à l'étude de l'« intelligence artificielle ».





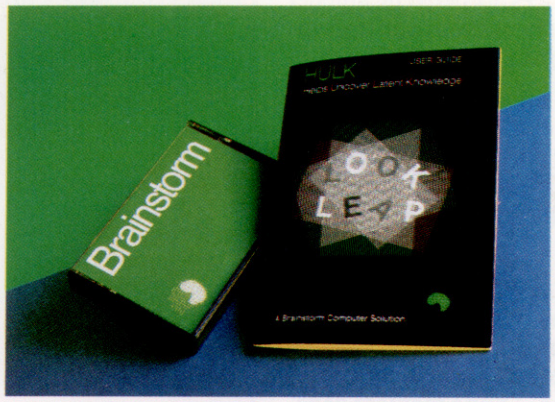
HULK à l'économie

Un système expert comporte traditionnellement un ensemble de règles de type IF ... THEN. Elles sont appliquées par le « moteur d'inférences ». Celui-ci les utilise en réponse aux questions de l'utilisateur qui passent par le module d'interrogation. Les connaissances sur lesquelles les règles reposent sont entrées dans le système par des experts, *via* le module d'acquisition. Avec HULK, l'ensemble des règles de décision est élaboré à partir du fichier principal de données qui contient les observations de l'utilisateur. Il existe deux programmes principaux : LOOK (examen), qui utilise des données d'essai pour élaborer des règles, puis des données de test pour déterminer leur pertinence ; et LEAP (conclusion), qui utilise le fichier de règles et teste les données pour produire un bilan sous la forme de probabilités. HULK a été surnommé le « système expert du pauvre ». (Cl. Liz Dixon.)

Un ingénieur « système expert » a pour fonction la conception de tels systèmes. Ces derniers constituent de véritables sommes de connaissances. Prenons l'exemple de la médecine : un médecin possède une connaissance approfondie des maladies, de leurs symptômes et de leurs traitements. Sa compétence tient au fait qu'il sait rattacher l'état de son patient aux connaissances qu'il a assimilées. Il détermine les symptômes dans son diagnostic, et évalue leur importance. Il s'appuie également sur son expérience propre et sur celle de ses collègues. Il lui faudra donc combiner au mieux ses connaissances avec les données fournies par l'observation du patient. La limite des capacités humaines prendra trois aspects : la somme considérable d'éléments à retenir, la difficulté de rattacher des données souvent complexes à ce qui est observé, l'incertitude du traitement à suivre. Les deux premiers facteurs sont par essence les attributs de l'ordinateur ; le troisième convient aux experts. Si l'ordinateur peut substituer l'analyse statistique au sens du diagnostic humain, et à ce qu'il comporte d'intuition, alors il est possible que ses capacités gigantesques de traitement surpassent le raisonnement humain.

De tels systèmes sont surtout indiqués pour des domaines où l'appréciation humaine est indispensable du fait de l'absence de théorie d'ensemble, comme pour la médecine. Ils peuvent également être d'un grand secours pour des domaines bien structurés mais complexes (par exemple la législation de l'impôt ou les barèmes de la Sécurité sociale).

Ces systèmes nécessitent l'enregistrement de connaissances, souvent sous la forme d'un ensemble de règles, et sont fréquemment appelés



Projet d'application

HULK a été initialement conçu par Richard Forsyth comme un projet d'application destiné à ses étudiants de troisième cycle de l'École polytechnique de North London. Bien que l'écriture du programme n'ait pris que deux semaines, sa mise au point, pour le rendre accessible à l'utilisateur, a ensuite demandé six mois. (Cl. Liz Heaney.)

pour cette raison « systèmes à base de règles » ou « systèmes fondés sur des connaissances ».

Les systèmes experts se sont révélés particulièrement efficaces dans plusieurs domaines. Ils surpassent parfois le diagnostic médical humain et servent utilement à la prospection géologique. Du fait de leurs réussites, ils quittent actuellement les laboratoires de recherche en intelligence artificielle pour trouver des applications dans tous les champs de l'informatique. Ils ont notamment une influence grandissante dans la conception des systèmes d'ordinateurs les plus performants. L'approche logicielle fondée sur les systèmes experts est en passe de remplacer le schéma :

données + algorithme = programme

par une méthodologie de la forme :

connaissances + inférences = système.

Cela constitue une étape importante dans l'évolution logicielle.

Ainsi, un système expert est fondé sur un ensemble de connaissances, ou *corpus*. Il existe en outre trois éléments essentiels : le *moteur* ou *générateur d'inférences,* qui crée de nouvelles règles pour l'interprétation des données à partir des règles existantes et des données; le *module d'acquisition de connaissances,* par l'intermédiaire duquel le système acquiert de nouvelles connaissances à partir de sa propre expérience et de celle des experts humains; et l'*interface utilisateur,* qui permet à des non-experts d'accéder au système.

Envisageons les deux modules fondamentaux, la base de connaissances et le moteur d'inférences.

Une base de connaissances comporte des énoncés et des règles. Les énoncés peuvent être modifiés rapidement, lors de la consultation du système. Les règles constituent une information à plus long terme servant à savoir générer les énoncés ou les hypothèses, à partir de l'acquis. En quoi cela est-il différent d'une base de données conventionnelle? La base de connaissances est davantage créative. Les éléments d'une base de données sont passifs. Leur présence est uniquement due au désir de l'utilisateur. Au contraire, une base de connaissances s'efforce de compléter les informations à la lumière de ce qu'elle « sait » déjà.

Les règles de la forme IF .... THEN (connues comme « règles génératrices ») peuvent amener à énoncer une probabilité, notamment du type « facteur 1 ». Par exemple :

> *Si* (IF) l'équipe jouant sur son terrain a perdu le dernier match chez elle, *et* (AND) si l'équipe en visite avait gagné le dernier match chez elle par deux buts, *alors* (THEN) la probabilité que l'équipe en visite marque un but est multipliée par 1,008.

Ces règles ne figurent pas en code programme. Ce sont des données en provenance du moteur d'inférences, qui constitue un interpréteur de haut niveau. Il existe deux méthodes d'inférence, la déduction et l'induction. D'une manière géné-

Prévisions météo

Les fichiers de données **HULK** sont créés par l'utilisateur en tant que fichiers programmes BASIC. Ils sont ainsi d'un accès facile et sont aisément éditables. Le fichier utilisé ici est typique. Nous « lançons » l'exécution de **HULK** par l'intermédiaire de ce fichier. Le système nous a demandé une hypothèse à propos des données : nous avons ainsi indiqué :

DEMAIN = 1

Cela signifie que nous nous intéressons aux échantillons statistiques dont la dernière variable a pour valeur 1. En d'autres termes, les jours où il a plu le lendemain. Nous voulons utiliser **HULK** pour prédire le temps de demain à partir du temps d'aujourd'hui. Nous sommes ensuite à même d'entrer dans l'ordinateur des règles à tester. Chacune d'entre elles se rapporte au fichier de données, et sa véracité est établie tant isolément que par rapport aux règles précédentes.
HULK donne son avis sur l'incorporation de la règle, mais la décision revient à l'utilisateur.
Nous avons en peu de temps mis au point trois règles susceptibles de prédire la pluie pour le lendemain :
1° s'il pleut pendant plus longtemps que 2 minutes aujourd'hui,

ET

2° si le soleil brille moins de 3 heures 30 minutes aujourd'hui,

ET

3° si la température maximale de demain n'excède pas la température minimale d'aujourd'hui de plus de 6 °C,

ALORS

vous pouvez à 83 % être sûr de vos prédictions en annonçant la pluie pour le lendemain (à supposer bien sûr que l'échantillon du mois soit représentatif).

Bien qu'il s'agisse d'un programme BASIC, il n'est pas destiné à être exécuté par RUN, mais sert uniquement à stocker des données.

```
1 METEO,30,5
```

La ligne 1 décrit le fichier : son nom (METEO), le nombre d'échantillons de données (30), et le nombre de types de données (5) par échantillon. Le fichier comporte les données météo pour un mois donné à Paris. Chaque jour comprend : les minima et les maxima (températures en degrés Celsius), les chutes de pluie (en millimètres), le nombre d'heures d'ensoleillement, et une variable booléenne qui prend la valeur 1 s'il a plu le jour suivant et la valeur 0 dans le cas inverse.

```
100 TEMPMINI
200 TEMPMAXI
300 PLUIES
400 SOLEIL
500 DEMAIN
```

Les lignes 100-500 donnent des noms de variables aux données de chaque échantillon.

```
1001 D01 ,54 , 110 , 175 , 32 , 1
1002 D02 ,42 , 125 , 041 , 62 , 1
1003 D04 ,76 , 112 , 077 , 11 , 1
1004 D04 ,27 , 105 , 018 , 43 , 0
1005 D05 ,30 , 120 , 000 , 95 , 0
1006 D06 ,44 , 106 , 000 , 55 , 0
1007 D07 ,48 , 094 , 000 , 51 , 1
1008 D08 ,68 , 092 , 055 , 48 , 1
1009 D09 ,64 , 102 , 048 , 41 , 1
        --------"------
1028 D28 ,58 , 154 , 000 , 20 , 1
1029 D29 ,67 , 088 , 064 , 42 , 0
1030 D30 ,45 , 096 , 000 , 68 , 1
```

Les lignes 1001 à 1030 contiennent les données, à raison d'un échantillon par ligne de programme. Chaque ligne débute par une étiquette qui identifie l'échantillon. Remarquez que les valeurs de HULK doivent être des nombres entiers, ce qui suppose que toutes les données soient multipliées par 10 pour conserver la précision.

La ligne 1001, par exemple, sera lue en réalité :

```
1001 D01,5.4,11.0,17.5,3.2,0.1
```

DEMAIN = 1

DEMANDE DE LA RÈGLE **1**

LA RÈGLE EST : **PLUIES > 20**

TABLE DE CONTINGENCE	RÉUSSITE	ÉCHEC
RÈGLE VRAIE	**12**	**2**
RÈGLE FAUSSE	**5**	**11**

TAUX DE RÉUSSITE = **76,3 %**

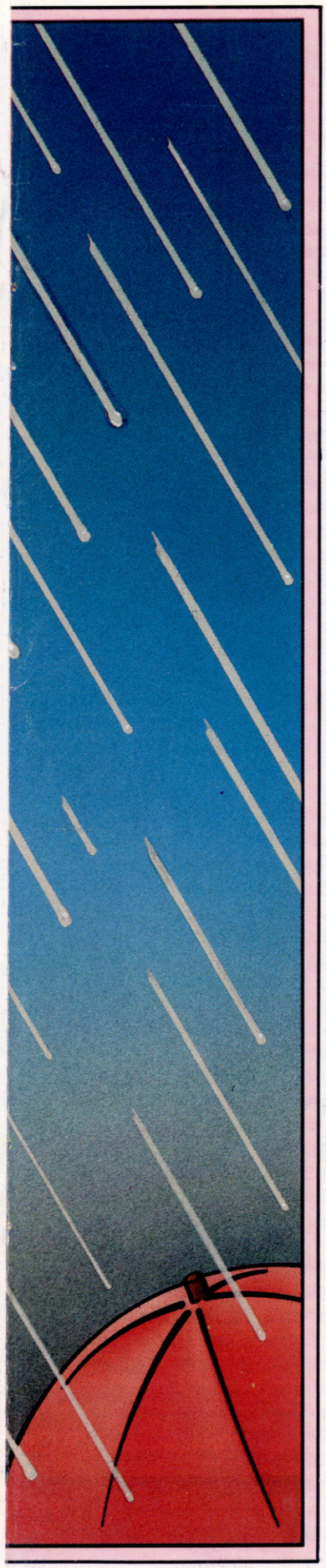
NOMBRE DE BITS PAR ÉCHANTILLON

AVANT	**PLUIE > 20**	**1,00**
APRÈS	**PLUIE > 20**	**0,85**

VOUS POUVEZ GARDER LA RÈGLE

LA RÈGLE A ÉTÉ AJOUTÉE A L'ENSEMBLE DES RÈGLES

Nombre de bits par échantillon
Message indiquant le nombre d'échantillons de données contribuant à la réussite de la règle. (Cl. Mike Brownlow.)



rale, on peut dire que la première déduit des hypothèses à partir des données, et que la seconde vérifie des hypothèses par les données. L'induction revient à un questionnement du type « que conclure si... ». La déduction est plus catégorique et tend à vérifier une affirmation.

La plupart des systèmes experts efficaces utilisent les deux méthodes. Qu'il s'agisse d'induction ou de déduction, les données resteront une matière première mal connue. Les spécialistes en informatique se sont efforcés sans grand succès de faire cadrer le monde avec la structure rigide de l'ordinateur. Les systèmes experts permettent enfin de traiter la réalité dans sa complexité, sans passer par une forme d'abstraction voulue par un système de données.

Il existe des formes variées de logiques. Certaines, dont la logique utilise des facteurs de certitude ou une fourchette de valeurs, confrontent l'affirmation « si "X" est vrai alors "Y" l'est aussi » à l'inférence statistique de probabilité « si "X" est vrai 65 fois sur 100 alors "Y" a entre 50 et 70 % de chances de l'être également ». De nombreux schémas ont été essayés et il semble curieusement qu'ils sont tous pertinents. Une explication possible serait que l'organisation des connaissances est plus importante que les valeurs numériques associées.

La plupart des bases de connaissances comportent des systèmes logiques redondants permettant de tirer les conclusions justes à partir de chemins différents. Les nombres mesurant le degré de vérité ne servent qu'à choisir entre plusieurs ensembles de valeurs.

Il existe maintenant sur le marché des progiciels destinés à faciliter l'approche « systèmes experts » de la conception des systèmes eux-mêmes. Jusqu'à présent, seul HULK (*Helps Uncover Latent Knowledge,* « mettre en évidence les connaissances latentes ») est accessible à l'amateur de micro-informatique.

HULK permet à l'utilisateur de se construire un ensemble de règles de décision et de les tester. Ces dernières peuvent ensuite être utilisées pour faire des prévisions ou à des fins de classification.

Vous trouverez ici plusieurs exemples types, chacun portant sur plusieurs variables. Ils vous aideront à mettre en évidence des schémas organisationnels pour prévisions.

Supposons par exemple qu'un agriculteur tienne à jour, sur micro-ordinateur, l'état des paramètres suivants pour une culture de betterave à sucre : hauteur et couleur de la feuille. Il sera à même de connaître la proportion de betteraves attaquées avant la récolte. Le système pourrait aboutir à diagnostiquer les maladies de cette plante selon l'état de ses feuilles. Ultérieurement, il pourrait permettre de sélectionner les espèces à moindre risque, et d'améliorer ainsi les rendements attendus pour la récolte suivante. L'agriculteur pourra très bien ne jamais connaître les règles que le système applique.

Prenons maintenant comme exemple les résultats d'une saison de football. Vous voulez créer un système qui vous permette de classer les équipes selon leurs performances passées et leurs caractéristiques, afin de faire des pronostics. Les données nécessaires, tels les derniers résultats de l'équipe qui reçoit, les résultats précédents dans la même épreuve ou le classement relatif des équipes, sont facilement accessibles. La confrontation et l'étude de ces données sont quant à elles trop complexes pour être menées à bien sans système expert. Ainsi, comme nous le voyons, il existe de nombreuses applications, pourvu que l'on ait des données.

Le progiciel HULK consiste en deux programmes principaux : LOOK (*Logical Organizer of Knowledge,* « Organisation logique des connaissances » — examen), et LEAP (*Likelihood Estimator and Predicator,* « estimation des probabilités et prévisions » — conclusion). Ils permettent à l'utilisateur de créer un ensemble de règles à partir d'un fichier de données d'observation sur disque (par exemple les résultats des matches précédents et les facteurs de décision). Ces règles sont alors appliquées à un autre fichier de données, incomplet celui-ci (par exemple les facteurs de décision pour le prochain match), afin d'établir des prévisions. L'ensemble des règles constitue la base de connaissances. Cette dernière rend compte des connaissances du système sur les données. Avec HULK, il s'agit de règles de décision fondées sur le calcul des probabilités.

Pour utiliser le module LOOK, il vous faut un ensemble de données, deux de préférence (vous pouvez à cette fin diviser un corpus important). Le premier ensemble de données s'appelle « données d'essais », et est destiné à essayer des règles. Le deuxième, appelé « données de test », doit établir la pertinence des règles sur des données non évidentes.

Une fois les données sur fichier, LOOK propose de nouvelles règles, une à la fois. Elles sont soumises à l'ensemble dit « données d'essais » et HULK évalue leur apport au système. L'utilisateur décide de les garder ou non, sur avis du système expert.

LOOK ne constitue pas un programme d'enseignement, mais plutôt un filtre qui permet de procéder en devinant ou en supposant. Il s'agit en quelque sorte d'un programme d'apprentissage sans algorithme. L'utilisateur apprend à maîtriser la question, et l'ordinateur effectue les tâches de comparaison, d'évaluation, et de test des données.

Ayant créé les règles avec LOOK, LEAP sert à les appliquer à des échantillons dont on ne sait rien. Les règles sont combinées pour obtenir une seule estimation pour chaque échantillon. LEAP fournit la liste des échantillons statistiques dans l'ordre de leur degré de probabilité de réalisation.

Cette dernière, ou « conclusion », dépend de la nature de l'échantillon de données (pour les exemples donnés plus haut, « rendement à l'hectare » et « score »). HULK fournit donc le module d'acquisition et celui de la base de connaissances, et laisse à l'utilisateur le soin de gérer le moteur, ou générateur d'inférences.

Lecteurs de disque

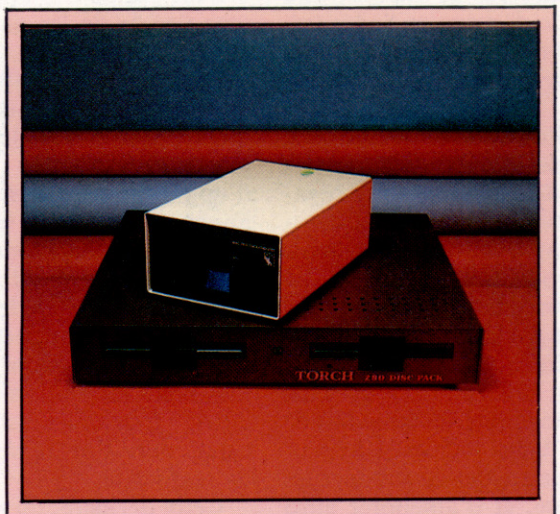
Acorn a fourni tous les détails nécessaires sur le système d'exploitation de son lecteur (Disk Filling System/système de remplissage du disque), et son contrôleur disque.

Le dieu du chaînage
Hobbit est un système de lecteurs à bandes pour le Micro BBC. Étant entièrement sous contrôle logiciel, toutes les fonctions de lecture, d'avance rapide, de rembobinage sont assurées automatiquement.

Ces lecteurs sont tous au standard disquette 5 pouces, et vont du lecteur relativement abordable en simple face, simple densité, au lecteur sophistiqué et coûteux « double face ». Des lecteurs 8 pouces seront également bientôt disponibles. Malheureusement les micros BBC ne comportent pas en standard le système de remplissage de disque. Il doit être acheté séparément. Il est fourni sous forme d'un composant mémoire morte venant dans un support sur la plaque à circuit imprimé. D'autres fabricants offrent des composants systèmes de remplissage de disque pour moins cher, mais la différence n'est pas significative.

Du fait que le système réside en mémoire morte, la plupart des commandes de disque ne nécessitent pas de mémoire interne pour exécuter leurs instructions. L'extension du système est assurée par des commandes et des routines fournies en tant qu'utilitaires sur une disquette livrée avec le lecteur. Ces derniers sont propres au type du lecteur et concernent le formatage et le contrôle du système.

Les lecteurs sont reliés au-dessous de l'ordinateur par un câble-nappe à 34 canaux. Ce câble véhicule toutes les données du disque. Un composant contrôleur disque de type 8271 gère les lecteurs en convertissant les données sur 8 bits en mode parallèle en provenance de l'ordinateur, en une sortie de type série vers le lecteur, et vice versa. Quatre types de disque sont disponibles : 40 pistes (ou 80 pistes) simple face ou 40 pistes (ou 80 pistes) double face. Tous les disques sont en simple densité. Chaque piste est divisée en 10 secteurs de 256 octets, soit un total de 100 K par face pour un format 40 pistes, et 200 K par face pour 80 pistes.



Lecteurs de disques BBC
Un lecteur de disques est essentiel pour le micro-ordinateur BBC. Les deux lecteurs présentés ici sont parmi les plus répandus pour le Model B, l'Acorn 100 et le Torch Z80. Avant d'être utilisés, ils doivent être interfacés *via* la mémoire morte du système d'exploitation. (Cl. Ian McKinnell.)

Les lecteurs sont identifiés par un numéro, 0 pour une unité de disque avec un seul lecteur, et 1 pour le second lecteur d'une unité de disque à deux lecteurs. Pour les lecteurs qui utilisent les deux faces du disque, chaque face est considérée comme une unité de disque distincte. Elles sont alors numérotées 0 et 2. Un second lecteur double face assignera les numéros 1 et 3 à ses deux faces.

Chaque catalogue d'un disque (ou d'une face) prend 2 K et se trouve sur les deux premiers secteurs de la première piste. Le catalogue contient l'information relative au nom et aux identificateurs du disque. Il peut comporter jusqu'à vingt-sept répertoires. Un répertoire est constitué de la liste des fichiers qu'il réunit à un titre quelconque. Il n'est pas tenu de table de disponibilité des secteurs (Block Availability Map). A la place, il y a la commande *COMPACT qui détecte tous les vides laissés par la suppression de fichiers, et réordonne les fichiers existants séquentiellement. Les espaces libres sont reportés après le dernier fichier.

Les programmes et les fichiers de données peuvent être stockés sur disquette de la même manière que sur bande. Le système de remplissage de disque transfère toutes les commandes de gestion de cassette.

A l'image de l'excellent BASIC BBC, le système de remplissage de disque BBC est très puissant pour manipuler des données. Il comprend en outre de nombreuses routines habituellement considérées comme inaccessibles aux micros. Ces dernières sont de surcroît faciles à utiliser. La structure du système permet une gestion efficace des disques et le transfert des données est rapide. Selon les fournisseurs des lecteurs, cette vitesse prendra approximativement cinq secondes pour charger un fichier programme de 20 K. L'inconvénient du système, mis à part son coût relativement élevé, est qu'il ne peut stocker que trente et un noms de fichiers par face de disque. Sachant que la capacité maximale est de 200 K et que le système de répertoires est très souple, cela représente une restriction dommageable.

Commandes disque BBC

Chaque fois qu'il faut préciser le fichier relatif à une commande, la spécification complète sera :

COMMAND:DV.DR.NOM

COMMAND est la commande de gestion de fichier demandée; DV (DriVe) est le numéro de lecteur (0-3); DR (DirectoRy) est l'identificateur de répertoire (A-X ou $); NOM se rapporte au nom du fichier et ne peut comporter que 7 caractères ('#','*','.', et ':' sont interdits). En l'absence de numéro de lecteur et d'identificateur de répertoire, le lecteur par défaut sera 0, et l'identificateur par défaut, $. Les identificateurs par défaut peuvent être changés avec les commandes *DRIVE,*DIR et *LIB. Le système de remplissage du disque permet également l'utilisation des caractères jokers '#' et '*'.

Pour certaines commandes, ils peuvent servir à spécifier tous les lecteurs, tous les répertoires, ou encore tous les noms de fichiers commençant par une même lettre. Dans les explications suivantes, la spécification ci-dessus sera notée <FSP> (File SPecification).

Le système de remplissage du disque propose les commandes suivantes de gestion de disque :

***FORM40, *FORM80 et *VERIFY**
Stockées sous forme de commandes utilitaires sur un disque livré avec les unités de disques. Elles formatent soit un disque 40 pistes, soit un disque 80 pistes, et elles indiquent si l'opération de formatage s'est bien passée.

ACCESS
*ACCESS<FSP>L protège le fichier contre toute destruction ou contre un effacement dû à une tentative d'écriture. *ACCESS<FSP> retire la protection.

***BACKUP, *DESTROY et *ENABLE**
*BACKUP DVsource DVdestination copie la totalité du disque présent dans le lecteur « source », sur le disque du lecteur « destination ». Ce dernier est dit « recouvert en écriture », donc effacé. *DESTROY<FSP> supprime un fichier. Si <FSP> comporte des jokers, plusieurs fichiers peuvent être détruits à la fois. BACKUP et DESTROY sont si radicaux dans leurs effets qu'une commande ENABLE doit avoir été préalablement passée (permission).

***BUILD<FSP>**
Crée un fichier ASCII aux spécifications entrées au clavier jusqu'à la frappe de la touche ESCAPE.

***CAT DV**
Affiche le catalogue du lecteur spécifié.

***COMPACT DV**
Déplace et réunit tout l'espace libre du disque spécifié, à la fin du dernier fichier.

***COPY**
Copie le(s) fichier(s) spécifié(s), en utilisant un joker pour nom de fichier, d'un disque sur l'autre.

***DELETE<FSP>**
Supprime le fichier spécifié du catalogue. On peut ensuite à nouveau écrire dedans.

***DIR DR**
Assigne le répertoire courant à celui spécifié. Tous les fichiers stockés après, avec *SAVE ou SAVE, seront assignés au répertoire spécifié.

***DRIVE DV**
Assigne le lecteur courant par défaut.

***DUMP<FSP>**
Affiche le fichier spécifié sous la forme d'un listage de valeurs hexadécimales.

***EXEC<FSP>**
Lit les données d'un fichier comme si elles provenaient de l'écran. Utile pour exécuter une séquence de programmes très courante. Les fichiers lus par *EXEC sont créés par *BUILD.

***HELP**
Concerne le fonctionnement du disque. *HELP DFS affiche une partie des commandes du système de remplissage de disque (Disk Filing System), ainsi que leur construction. *HELP UTILS affiche le reste.

***INFO<FSP>**
Affiche des informations sur le(s) fichiers(s) spécifié(s) (en utilisant des jokers) non affichés par *CAT : emplacement mémoire, adresse d'exécution, longueur en octets et emplacement du secteur.

***LIB:DV.DR**
Assigne à la bibliothèque le répertoire spécifié. Autorise la commande *FILENAME qui recherche le répertoire courant de la bibliothèque pour le programme code machine indiqué, le charge en mémoire pour l'exécuter immédiatement comme avec la commande *RUN.

***LIST<FSP>**
Affiche le fichier ASCII spécifié avec les numéros de lignes.

***LOAD<FSP>**
Replace en mémoire le fichier spécifié, sur les positions dont ils provenaient.

***OPT1**
Active un système de messages intervenant à chaque accès à un fichier, sur l'emplacement des informations transmises par *INFO. Ce dispositif est activé par *OPT 1 1 et inhibé par OPT 1 0.

***OPT4**
*OPT4 ou (SHIFT) BREAK change l'option d'autochargement du fichier spécifié courant à la mise sous tension. *OPT4 0 inhibe l'autochargement, *OPT4 1 charge par LOAD le fichier !BOOT, OPT 4 2 exécute par RUN !BOOT, et *OPT 4 4 exécute ce même fichier par EXEC.

***RENAME<anc.FSP> <nouv.FSP>**
Modifie le nom d'un fichier et met ce nom dans un autre répertoire. Ne peut déplacer des fichiers d'un lecteur sur un autre.

***RUN<FSP>**
Prend un fichier code machine en mémoire et l'exécute immédiatement. Sert pour des fichiers ne figurant pas dans la mémoire courante.

***SAVE**
Copie le bloc spécifié de la mémoire de l'ordinateur et l'écrit sur disque sur le lecteur et le répertoire courants. De syntaxe :

*SAVE«NOM» SSSS FFFF EEEE RRRR

ou

*SAVE«NOM» SSSS+LLLL EEEE RRRR

SSSS est l'adresse de début du bloc de la mémoire; FFFF, l'adresse de fin; EEEE, l'adresse d'exécution du programme stocké; RRRR, l'adresse de transfert du programme; et LLLL, la longueur du fichier en octet (alternative de FFFF). Tous les nombres sont en hexadécimal. RRRR et EEEE peuvent être omis, auquel cas les adresses de rechargement et d'exécution seront SSSS par défaut.

***SPOOL<FSP>**
Ouvre le fichier spécifié afin de recevoir les informations affichées en tant que fichier texte. Permet à un programme BASIC d'être stocké sous la forme d'un fichier ASCII plutôt que d'être transcrit en code machine.

***TITLE«NOM DISQ.»**
Change le nom du disque dans le lecteur courant pour celui spécifié.

***TYPE<FSP>**
Affiche un fichier ASCII sans numéros de lignes.

***WIPE<FSP>**
Identique à *DESTROY à la différence près que *ENABLE n'est pas nécessaire ici.

Banc d'essai

Le stade final de tout montage de circuit consiste à le tester. Un simple testeur d'intensité ne vous coûtera que quelques francs, et un multimètre sophistiqué restera tout à fait abordable.

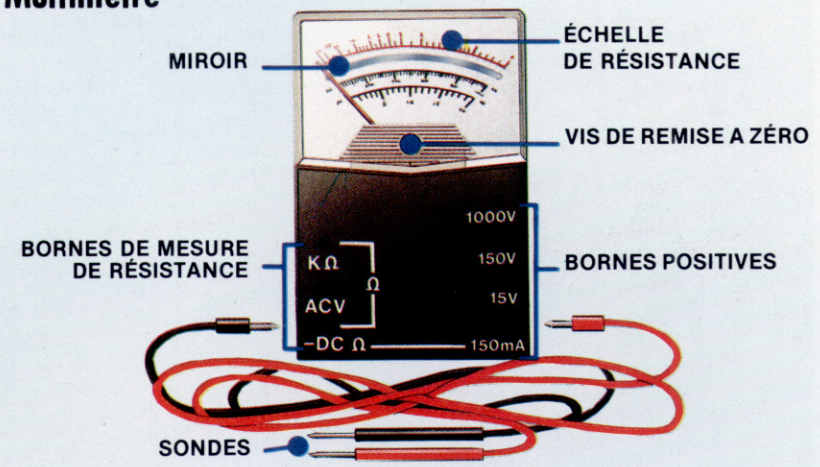
Pour tester un circuit comportant plusieurs composants, il faut d'abord s'assurer de l'efficacité des soudures. Si elles ont été faites à température trop basse, elles auront peut-être un aspect extérieur satisfaisant, mais ne serviront à rien. Une légère pression peut vous permettre de vous en rendre compte; si la soudure a été faite correctement, elle ne vous restera pas dans les mains. Sinon, mieux vaut s'en rendre compte tout de suite.

Il vous faut maintenant tester le circuit lui-même. L'outil le plus simple dans ce cas est vendu dans les boutiques d'accessoires de voiture, ou de composants électroniques. On l'appelle parfois « vérificateur d'allumage », et il permet de déterminer si les points de contact du distributeur sont ouverts ou fermés.

Pour ce que nous voulons faire, la pince crocodile installée à une extrémité de l'appareil n'est pas des plus utiles; nous y fixerons donc, à l'aide de ruban isolant, la pointe d'un fer à souder, qui fera office de sonde. Il est d'ailleurs très simple de construire soi-même ce type d'engin, à l'aide de trois piles sèches de 1,5 V, d'une ampoule de 3 V et d'un peu de fil électrique.

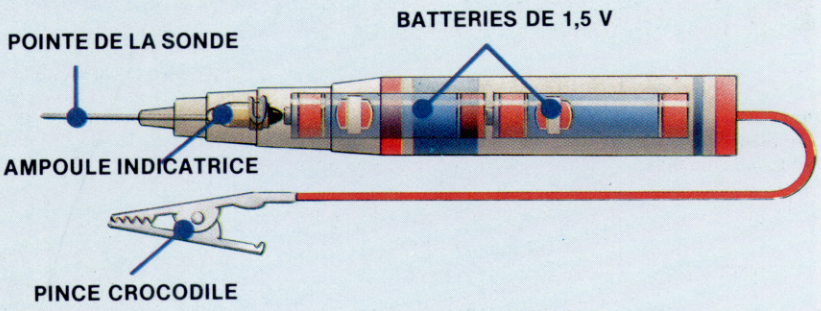
Mais il est sans doute préférable de se servir d'un petit multimètre (qu'on appelle aussi « contrôleur universel »). C'est un dispositif qui peut tester non seulement l'intensité, mais aussi la résistance. L'ohm, ainsi nommé d'après le physicien allemand Georg Ohm (1787-1854) qui découvrit le phénomène, est l'unité de résistance; celle-ci est fonction de la section du fil électrique parcouru par un courant. Mais on peut également l'introduire artificiellement grâce à des composants eux aussi appelés résistances. En ce qui nous concerne, une soudure proprement faite n'offrira qu'une résistance négligeable au faible courant fourni par nos appareils, et leurs systèmes de contrôle fonctionneront à plein : l'aiguille du cadran sera fortement déviée, la petite ampoule s'allumera sans ambiguïté. Toute réaction moins marquée est l'indice probable d'une source médiocre, qu'il faudra refaire. Le multimètre a encore deux fonctions supplémentaires : mesurer le courant et le potentiel électrique.

Multimètre

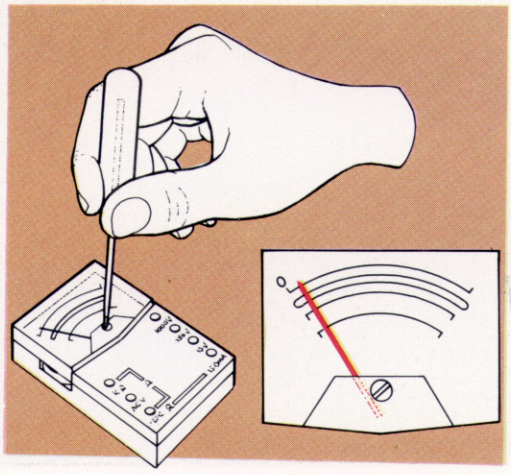


Les multimètres, qui testent l'intensité et la résistance (mesurée en ohms), l'intensité du courant (mesurée en ampères ou A) et la différence de potentiel (mesurée en volts ou V), sont de prix très variable. Certains sont très bon marché, d'autres beaucoup plus sophistiqués valent des milliers de francs. Il n'existe pourtant que deux méthodes de présentation de leurs résultats : analogique ou numérique. En général, les instruments à bobine mobile affichent leurs données en déplaçant une aiguille sur un cadran, en fonction de l'augmentation ou de la baisse de la valeur qu'ils mesurent; ils sont beaucoup moins chers que les modèles numériques.

Testeur de circuit



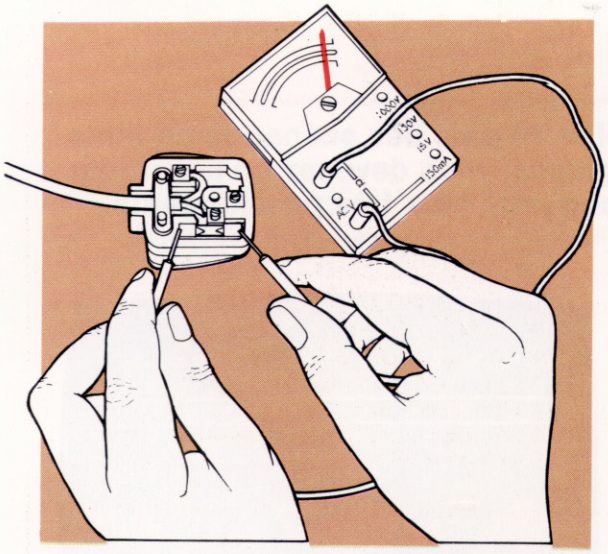
Remise à zéro



Remise à zéro

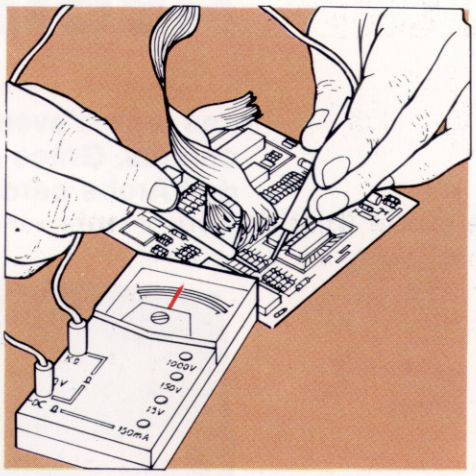
Les appareils de mesure de type analogique, qui utilisent une aiguille se déplaçant au-dessus d'un cadran, doivent pouvoir être ajustés. Une vis montée sur le pivot de l'aiguille y pourvoit. Un petit miroir placé derrière l'aiguille permet d'éviter des erreurs de réglage dues à une mauvaise lecture. Le circuit de mesure de la résistance doit également pouvoir tenir compte des variations et des irrégularités diverses. Lui aussi est à régler de temps en temps pour éviter les erreurs.

Continuité

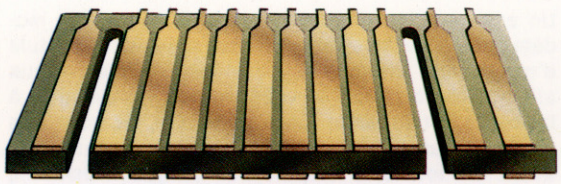


Continuité et résistance
Un fusible — comme il en est installé sur toutes les prises à trois broches de 13 A — est un dispositif délibérément conçu pour provoquer une interruption de courant. Il fond et entraîne la rupture du circuit en cas de surcharge; lorsqu'un montage refuse de fonctionner et qu'on n'a pas de testeur sous la main, un remède simple consiste à remplacer l'ancien fusible par un nouveau. Mais si la cause de la panne n'est pas là? Dans ce cas un testeur de circuit aura tôt fait de nous donner la réponse. Un multimètre peut aussi être utilisé pour donner la valeur d'une résistance.

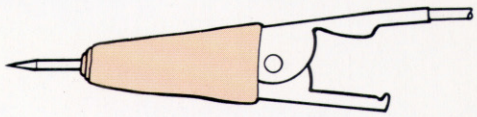
Résistance



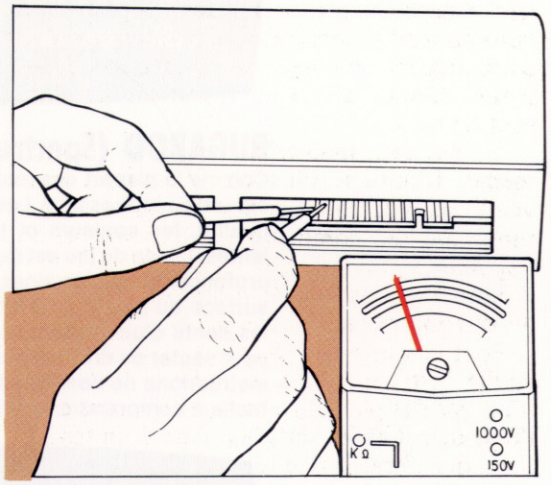
Connecteurs
Les connecteurs plats, comme celui qui est représenté ici, permettent de relier l'ordinateur à ses périphériques. Certains micros ne disposent que d'un seul de cet élément; d'autres en ont plusieurs. Les connecteurs plats ne sont jamais qu'un type de port entrée/sortie parmi d'autres; mais ils ont l'avantage de faire partie intégrante du circuit imprimé.

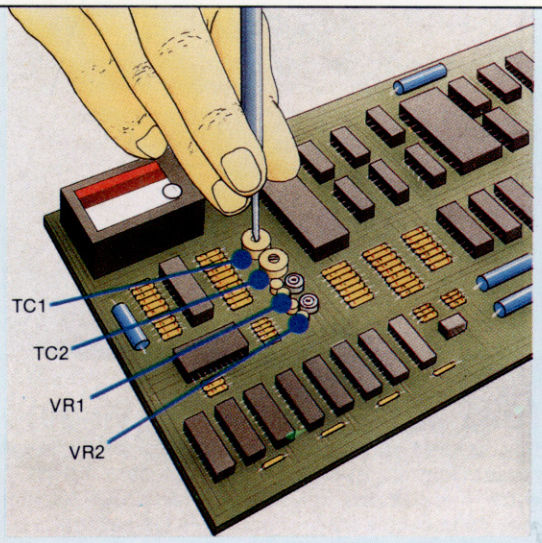


Pince crocodile
Peu précise, la pince crocodile peut être transformée en sonde plus fine, en plaçant entre ses mâchoires la pointe d'un fer à souder, que l'on fixera grâce à un ruban isolant.



Voltage
Tester la source d'alimentation du connecteur plat de votre ordinateur est un bon moyen de voir comment fonctionne le multimètre quand on s'en sert comme testeur de voltage. Cherchez dans votre manuel utilisateur les broches de 0, + 5 et + 9 V. Mettez en contact la première avec la borne négative de l'appareil, et l'une des deux autres avec la borne positive. Le résultat inscrit sur le cadran vous indiquera la précision de la régulation de tension et du contrôle.





Réglage du Spectrum

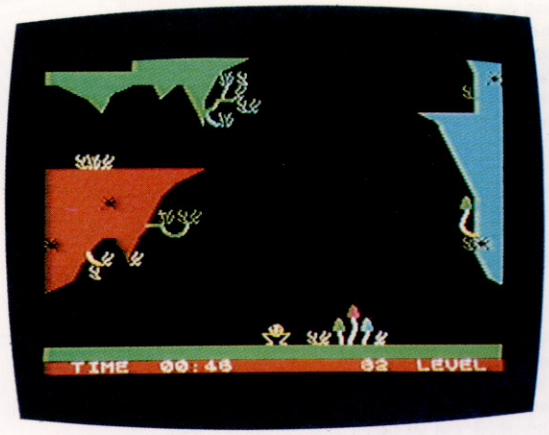
Comme nous l'avons fait remarquer, les possesseurs de premiers modèles du Spectrum peuvent remédier à la détérioration de l'image TV. Les deux rhéostats (VR1 et VR2) signalés ici contrôlent respectivement le rapport rouge-vert et le rapport jaune-bleu. TC1 et TC2 sont deux condensateurs ajustables qui règlent l'intensité de la couleur et la netteté des caractères.
On ouvre le boîtier du Spectrum en enlevant les cinq vis placées tout en dessous de l'appareil. Ne négligez pas la mise en garde relative à l'annulation de la garantie.
La version 3 du Spectrum ne permet pas d'effectuer ce réglage — ni d'ailleurs quelque réglage que ce soit. On la reconnaît à son dissipateur de chaleur, une épaisse plaque d'aluminium située à gauche du connecteur plat.

ATTENTION

La garantie de votre ordinateur, si elle est toujours en vigueur, peut être annulée si quelqu'un d'autre que l'importateur, ou son représentant agréé, ouvre le boîtier

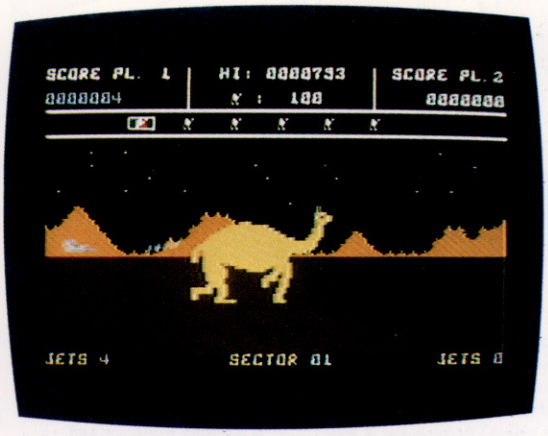
Aventures

Les jeux d'aventures s'inspirent de plus en plus des scénarios de films connus. Grâce aux progrès technologiques, vous devenez un aventurier de l'Arche perdue. En attendant, voici quatre jeux très passionnants.



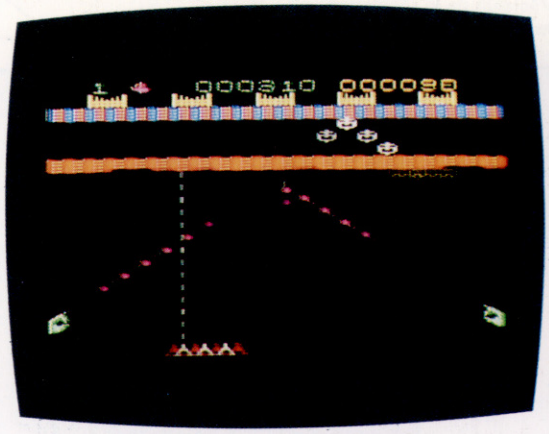
BUGABOO (Spectrum)

Comme la plupart des jeux sur Spectrum, BUGABOO est un jeu intéressant. Les graphiques sont de bonne qualité, les couleurs et les animations sont excellentes. L'idée du jeu est de faire sauter la mouche des profondeurs souterraines de la caverne jusqu'à la surface du sol, le plus vite possible, tout en évitant les dents d'un dragon peu sympathique. La mouche peut sauter de droite à gauche et de haut en bas. Les instructions ne sont pas très claires, mais le jeu est facile à comprendre après quelques essais.



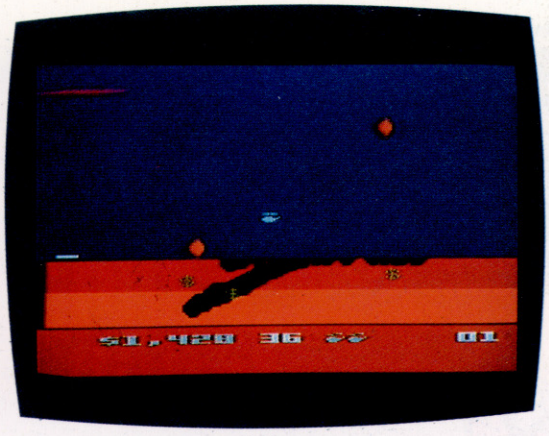
ATTACK OF MUTANT CAMELS (CBM 64)

Un excellent jeu d'arcades, dans la lignée des modèles du genre. Vous êtes attaqué dans votre module d'exploration interstellaire par des chameaux aux allures très bizarres et vous pouvez les détruire à l'aide de votre canon. Mais attention, ils sont très résistants et vous devez les toucher plusieurs fois avant de les faire disparaître. Un jeu irrésistible conçu par un as de la programmation.



FOUR GATES TO FREEDOM (Vic-20)

Des savants sont retenus prisonniers par de dangereuses fourmis guerrières de la planète Xzinos et votre mission est de les délivrer. Pour parvenir jusqu'à la cellule où sont enfermés les savants, vous devez franchir quatre niveaux différents de la planète Xzinos, fermés par quatre portes dont il vous faudra trouver les clés au terme de durs et farouches combats contre les fourmis.
En résumé, ce jeu se rapproche très nettement de la qualité des jeux d'arcades, dont on sait tout l'intérêt qu'ils suscitent.



BURIED BUCKS (Atari)

Vous êtes un pilote d'hélicoptère et il vous faut ramasser autant d'argent que possible, pour pouvoir faire marcher votre machine. De l'argent, il y en a sur cette planète, mais il est enfoui sous des tonnes de terre qu'il faut faire sauter à la dynamite! A première vue, cela paraît évident; mais, malheureusement, dès que l'argent est accessible, un avion surgit et vient remplir le trou que vous avez réalisé en le bombardant de gros rochers. Plus vous progressez, plus le jeu devient difficile. Sans parler des lacs et des rivières souterraines qui risquent de tout remplir!

Un ordinateur impersonnel?

IBM est le plus important constructeur d'ordinateurs et d'équipements de bureau du monde. La firme ne s'est pourtant aventurée sur le marché du micro-ordinateur qu'en 1981 avec le PC. Est-ce le succès?

Dans le domaine des matériels, les gros systèmes et les mini-ordinateurs d'IBM n'ont pas la réputation d'être très novateurs. Les logiciels et leur documentation (par ailleurs extrêmement précise et détaillée) pourraient sans doute être améliorés au niveau de la présentation et de la simplicité d'emploi. Mais « Big Blue », comme on l'appelle aux États-Unis, est connu pour la robustesse de ses produits, et l'IBM PC ne fait pas exception à la règle. Comme tous les matériels de la firme, il est aussi nettement plus cher que ses concurrents.

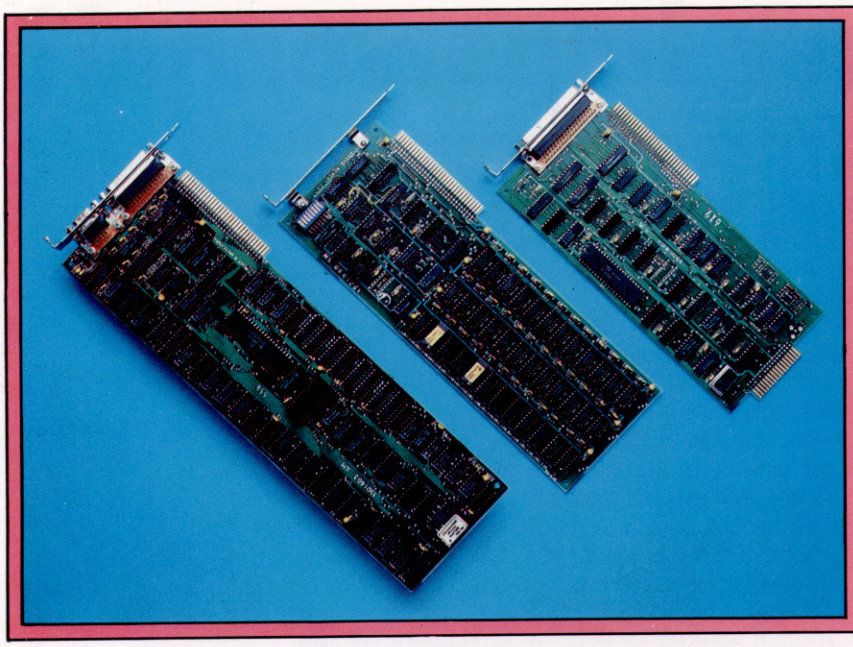
Pourtant cela n'empêche pas les ventes; la machine est vite devenue très populaire, en dépit d'un prix qui aurait pu paraître dissuasif. Honneur suprême : elle a été aussi copiée et contrefaite que l'Apple, et bien plus vite.

IBM explique qu'un coût aussi élevé tient à l'assistance technique fournie par la firme. Vous pouvez effectivement en bénéficier — si vous êtes prêt à payer 11,2 % du prix de l'appareil par an pour un contrat de maintenance. D'autres compagnies proposent leurs services à des tarifs encore plus onéreux (sans être forcément en mesure de procéder à des échanges rapides de matériels). Il faut vraiment être résolu à acheter les produits d'une aussi grosse maison qu'IBM.

Les caractéristiques de l'IBM PC n'ont rien d'exceptionnel. Il est construit autour d'un microprocesseur 8088, qu'on décrit comme un 16-bits, mais où les lignes d'adressage et de données sont multiplexées pour soulager des broches sur le circuit; aussi est-il assez peu rapide. En fait, par rapport à un 8-bits classique, la vitesse d'exécution ne s'accroît que de 25 %.

Le modèle de base ne donnera toute sa mesure qu'avec des extensions. En effet, il manque de mémoire (ou plutôt n'en a pas assez pour faire tourner des programmes complexes), et son système d'entrée/sortie est très limité. Les acheteurs se rendent vite compte qu'il leur

Un design ergonomique
La conception et l'apparence extérieure de l'IBM ont bénéficié de l'énorme expérience de la firme dans le domaine des ordinateurs et de l'équipement de bureau. Le tout est peu encombrant, très satisfaisant d'un point de vue ergonomique. Les trois constituants principaux — le clavier, le processeur et le moniteur — sont séparés, ce qui facilite la mise en place et l'utilisation. La version de base n'offre qu'un moniteur monochrome, mais une version couleur est disponible.
(Cl. Chris Stevens.)



Cartes d'extension
Dans sa version de base, l'IBM PC peut difficilement concurrencer des appareils comme l'Apple II, bien plus prestigieux. Mais avec l'adjonction (de gauche à droite) d'une carte graphique couleur, d'une extension mémoire et d'un contrôleur d'entrée/sortie très sophistiqué, il devient enfin un véritable ordinateur personnel orienté vers la gestion.

faudra au moins une carte multifonction supplémentaire, et elle coûte autant que bien des micro-ordinateurs.

Les possibilités graphiques sont impressionnantes, mais ne sont accessibles que par l'intermédiaire d'une carte graphique. Elle existe en deux versions (monochrome et couleur), et comprend des composants électroniques qui produisent le signal vidéo, ainsi qu'un vaste bloc de mémoire (qui gère les images produites à l'écran). Elle est contrôlée par l'unité centrale; mais un nouveau modèle existe, qui dispose de son propre processeur : celui-ci se charge lui-même de la mise à jour des écrans, et les opérations sont donc plus rapides. Mais tout cela est encore très cher.

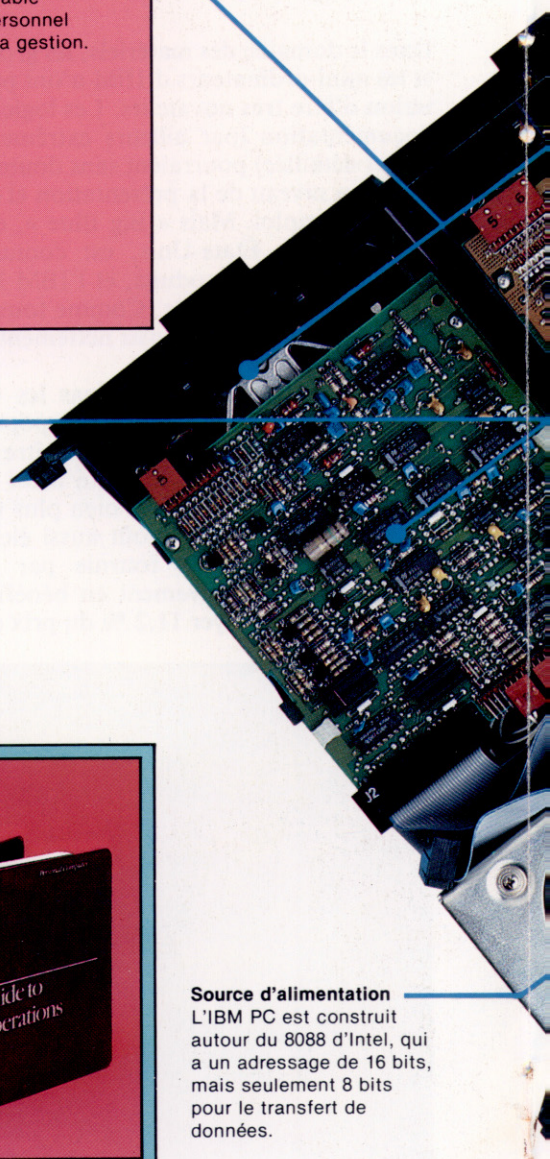
Le PC n'a que cinq ports d'extension, et il faut donc les utiliser à bon escient. Il s'ensuit qu'on ne peut se procurer de cartes bon marché, aux fonctions limitées; elles sont presque toutes larges, complexes, capables de faire bien des choses (souvent en même temps) et sont très onéreuses. L'IBM PC pouvant difficilement s'en passer, on fera bien de tenir compte de leur prix si l'on envisage d'en acheter un.

Il existe d'autres variantes de l'appareil — ainsi le modèle XT à disque dur — qui intègrent toutes ces facilités. Mais il faudra alors dépenser bien davantage, presque autant que pour le Lisa d'Apple, qui est d'une conception bien plus moderne.

Sacrifiant à la mode, IBM a même sorti une version portable du PC. Elle pèse 14 kg, le terme est donc un peu excessif... Les deux lecteurs de disquettes ont été remplacés par un seul (double face), et l'espace ainsi libéré accueille un moniteur ambre de neuf pouces. Le clavier, réduit et plus léger, se fixe à l'avant de la machine, placée dans un nouveau boîtier.

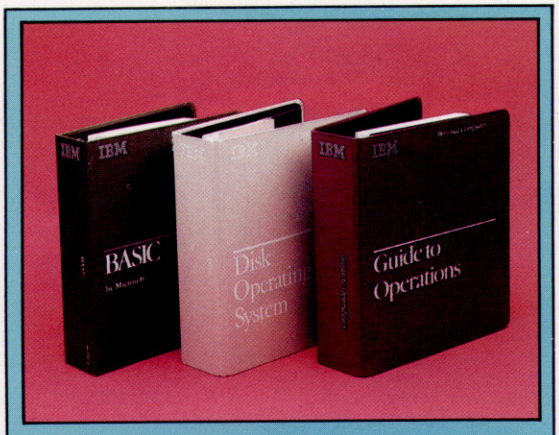
Ces remarques n'empêchent pas de faire planer sur ce produit un doute que la notoriété d'IBM ne peut accepter trop longtemps.

Lecteur de disquettes
La machine n'est vendue qu'avec un seul lecteur de disquettes (monoface, simple densité), mais on peut passer progressivement à la double face double densité.

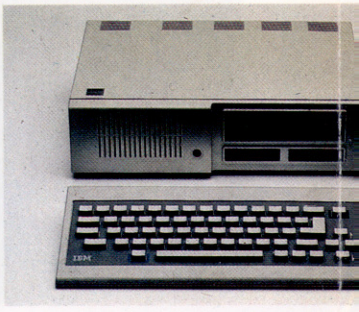


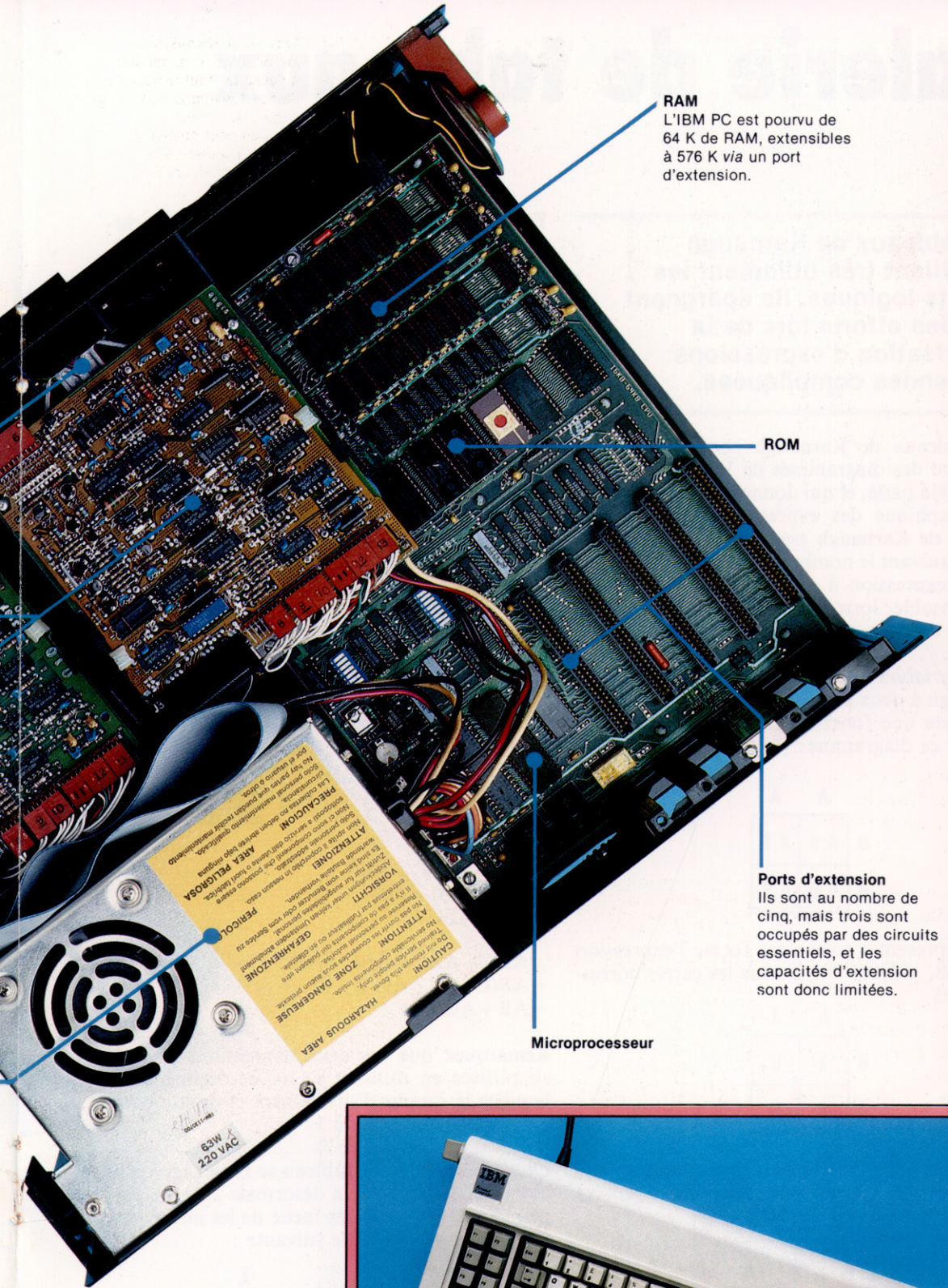
Circuits de contrôle disquette

Source d'alimentation
L'IBM PC est construit autour du 8088 d'Intel, qui a un adressage de 16 bits, mais seulement 8 bits pour le transfert de données.



Les logiciels
L'une des premières raisons d'acheter un IBM PC (et qui explique que bien des constructeurs du monde entier l'aient aussitôt copié dans les moindres détails) est l'immense choix de logiciels disponibles. Tous fonctionnent sous le contrôle du PC-DOS (système d'exploitation sur disquette) mis au point par Microsoft à partir du CP/M, mais d'autres possibilités peuvent être envisagées : ainsi le CP/M-86 ou le système P de l'UCSD. Les langages utilisables sont très nombreux : COBOL, PASCAL, FORTRAN, BASIC (entre autres). Les progiciels sont eux aussi très différents : traitements de texte, tableurs, bases de données, logiciels de gestion. L'IBM PC est également considéré aux États-Unis comme un ordinateur domestique, contrairement à ce qu'on pense ailleurs, et beaucoup de jeux ont été écrits pour lui.





RAM
L'IBM PC est pourvu de 64 K de RAM, extensibles à 576 K *via* un port d'extension.

ROM

Ports d'extension
Ils sont au nombre de cinq, mais trois sont occupés par des circuits essentiels, et les capacités d'extension sont donc limitées.

Microprocesseur

IBM PC

PRIX
****

DIMENSIONS
140 × 500 × 400 mm.

UNITÉ CENTRALE
8088 Intel.

CAPACITÉ MÉMOIRE ET VITESSE
RAM de 64 K, extensible à 576 K.
ROM de 40 K.
4,7 MHz.

CARACTÉRISTIQUES ÉCRAN
25 lignes de 80 caractères.

INTERFACES ET PORTS
Centronics parallèle et cinq ports.

LANGAGES DISPONIBLES
BASIC, et sous PC-DOS un choix comprenant COBOL, FORTRAN, etc.

CLAVIER
79 touches type machine à écrire.

DOCUMENTATION
De très haut niveau chez IBM et ses fournisseurs agréés. De qualité plus inégale chez les producteurs indépendants de logiciels.

POINTS FORTS
La version de base de l'IBM PC fonctionne très bien, et les extensions en font l'un des micro-ordinateurs les plus puissants du marché. Et c'est IBM qui le fabrique.

POINTS FAIBLES
Très cher quand on le compare aux nombreux compatibles apparus après sa sortie. Il est aussi assez complexe, et les logiciels et la maintenance sont plus onéreux que pour bien des machines plus simples.

Le Peanut
Le PC Junior d'IBM, vite surnommé « Peanut » avant même sa sortie, est une version extrêmement simplifiée du PC. L'innovation la plus intéressante est sans doute une communication par infrarouges entre le clavier et l'unité centrale, qui remplace l'habituelle liaison par câble.

Clavier
Comme on pouvait s'y attendre de la part d'un aussi gros fabricant de machines à écrire, de terminaux et d'appareils du même ordre, le clavier est virtuellement sans défaut. Il est séparé du reste de l'appareil, et l'utilisateur peut le placer comme il le désire. Très plat, il comporte cinq rangées de touches modelées, disposées de façon à former un secteur de la circonférence d'un vaste tambour.
(Cl. Chris Stevens.)

Galerie de tableaux

Les tableaux de Karnaugh simplifient très utilement les circuits logiques. Ils épargnent bien des efforts lors de la factorisation d'expressions booléennes compliquées.

Les tableaux de Karnaugh sont en fait une extension des diagrammes de Venn, dont nous avons déjà parlé, et qui donnent une représentation graphique des expressions logiques. Un tableau de Karnaugh peut prendre différentes formes suivant le nombre de variables que comporte l'expression à simplifier, mais se révèle indispensable lorsque ces variables sont au nombre de deux, trois ou quatre.

Deux variables : chaque carré d'un tableau de Karnaugh à deux variables (il y a $2^2 = 4$ carrés) représente une fonction ET, comme on peut le voir sur ce diagramme :

	A	$\overline{A}$
B	A.B	$\overline{A}$.B
$\overline{B}$	A.$\overline{B}$	$\overline{A}$.$\overline{B}$

Pour représenter sous cette forme l'expression $AB + A\overline{B}$, plaçons des 1 dans les carrés correspondants :

	A	$\overline{A}$
B	1	0
$\overline{B}$	1	0

Voici trois autres exemples, représentant respectivement les expressions $\overline{A}\overline{B}$, $AB + \overline{A}\overline{B}$ et $AB + \overline{A}B + A\overline{B}$:

	A	$\overline{A}$
B	0	0
$\overline{B}$	0	1

	A	$\overline{A}$
B	1	0
$\overline{B}$	0	1

	A	$\overline{A}$
B	1	1
$\overline{B}$	1	0

Trois variables. Dans ce cas, le nombre de carrés augmente d'un facteur deux ($2^3 = 8$ carrés), et un tableau à trois variables aura la forme suivante :

		A	$\overline{A}$	
B	C	A.B.C	$\overline{A}$.B.C	C
		A.$\overline{B}$.C	$\overline{A}$.$\overline{B}$.C	
$\overline{B}$		A.$\overline{B}$.$\overline{C}$	$\overline{A}$.$\overline{B}$.$\overline{C}$	$\overline{C}$
B		A.B.$\overline{C}$	$\overline{A}$.B.$\overline{C}$	

Voici deux expressions, $AC + \overline{A}B\overline{C}$ et $AB + \overline{A}C$, représentées sous cette forme :

	A	$\overline{A}$	
B	1	0	C
$\overline{B}$	1	0	
	0	0	$\overline{C}$
B	0	1	

	A	$\overline{A}$	
B	1	1	C
$\overline{B}$	0	1	
	0	0	$\overline{C}$
B	1	0	

$$\begin{aligned} &ABC + A\overline{B}C + \overline{A}B\overline{C} \\ &= AC(B + \overline{B}) + \overline{A}B\overline{C} \\ &= AC + \overline{A}B\overline{C} \end{aligned}$$

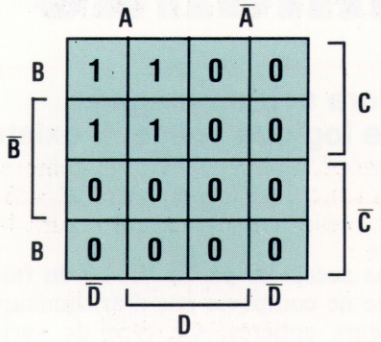
$$\begin{aligned} &ABC + AB\overline{C} + \overline{A}BC + \overline{A}\overline{B}C \\ &= AB(C + \overline{C}) + \overline{A}C(B + \overline{B}) \\ &= AB + \overline{A}C \end{aligned}$$

Remarquez que les deux expressions ont été simplifiées en utilisant la loi booléenne selon laquelle la somme d'un élément et de son complément ($\overline{A}$) est égale à 1.

Quatre variables. Le tableau se fait encore plus complexe (puisqu'il y a désormais $2^4 = 16$ carrés). Pourtant il est très facile de les interpréter par référence à la grille suivante :

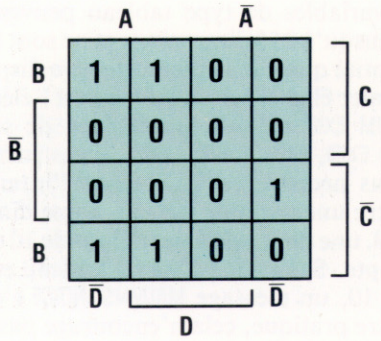
		A		$\overline{A}$		
B	C	A.B.C.$\overline{D}$	A.B.C.D	$\overline{A}$.B.C.D	$\overline{A}$.B.C.$\overline{D}$	C
		A.$\overline{B}$.C.$\overline{D}$	A.$\overline{B}$.C.D	$\overline{A}$.$\overline{B}$.C.D	$\overline{A}$.$\overline{B}$.C.$\overline{D}$	
$\overline{B}$		A.$\overline{B}$.$\overline{C}$.$\overline{D}$	A.$\overline{B}$.$\overline{C}$.D	$\overline{A}$.$\overline{B}$.$\overline{C}$.D	$\overline{A}$.$\overline{B}$.$\overline{C}$.$\overline{D}$	$\overline{C}$
B		A.B.$\overline{C}$.$\overline{D}$	A.B.$\overline{C}$.D	$\overline{A}$.B.$\overline{C}$.D	$\overline{A}$.B.$\overline{C}$.$\overline{D}$	
		$\overline{D}$	D		$\overline{D}$	

Voici un tableau de ce type, avec les simplifications appropriées :



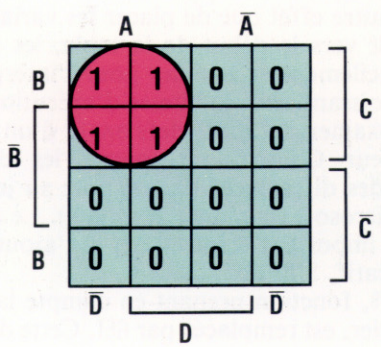
$$\begin{aligned}
&ABC\bar{D} + ABCD + A\bar{B}C\bar{D} + A\bar{B}CD \\
&= ABC(\bar{D} + D) + A\bar{B}C(\bar{D} + D) \\
&= ABC + A\bar{B}C \\
&= AC(B + \bar{B}) \\
&= AC
\end{aligned}$$

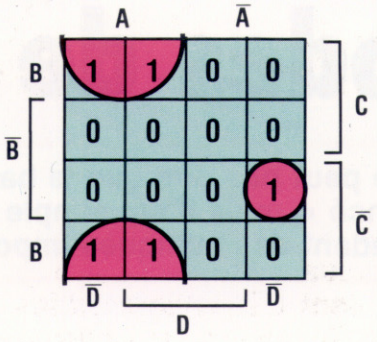
Autre exemple :



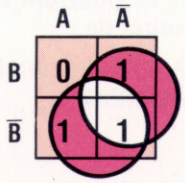
$$\begin{aligned}
&ABC\bar{D} + ABCD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D \\
&= ABC(\bar{D} + D) + \bar{A}\bar{B}C\bar{D} + AB\bar{C}(\bar{D} + D) \\
&= ABC + \bar{A}\bar{B}C\bar{D} + AB\bar{C} \\
&= AB(C + \bar{C}) + \bar{A}\bar{B}C\bar{D} \\
&= AB + \bar{A}\bar{B}C\bar{D}
\end{aligned}$$

Si par ailleurs on examine attentivement la disposition des 1 sur ces deux tableaux, on remarque qu'elle obéit à des règles. Dans le premier cas, tous les carrés comportant AC dans leurs expressions ont des 1 ; dans le second, la situation est la même pour tous les carrés AB. Cela laisse entrevoir qu'un moyen encore plus simple de réduire des expressions booléennes consiste à étudier de près un tableau de Karnaugh. Par exemple :





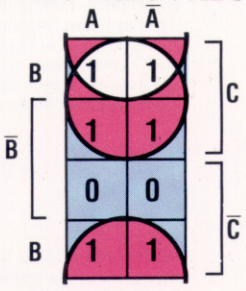
Avec un peu d'expérience il est possible de repérer des groupes de deux, quatre ou huit 1, et de les simplifier. Considérons ainsi l'expression suivante : $\bar{A}B + A\bar{B} + \bar{A}\bar{B}$.



Grâce à un tableau à deux variables, nous sommes en mesure d'identifier deux groupes de 1. Le premier représente les occurrences NON(B) et l'autre les occurrences NON(A), et toute l'expression peut être réduite à $\bar{A} + \bar{B}$. La loi de Morgan permet de surcroît d'aller plus loin, jusqu'à $\bar{A}.\bar{B}$. Est-il possible de parvenir plus directement à cette conclusion en inspectant le tableau ?

Un exemple plus ardu met en jeu une expression à trois variables :

$$ABC + A\bar{B}C + \bar{A}BC + AB\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$$



Le groupe de quatre 1 situé au sommet représente tous les cas possibles dans lesquels C est vrai. Les deux 1 tout en haut et les deux 1 du bas représentent tous les cas possibles dans lesquels B est vrai. L'expression peut donc être réduite à $B + C$.

Lors de notre prochain cours, nous approfondirons l'emploi des tableaux de Karnaugh dans la simplification des expressions booléennes. Ensuite nous montrerons comment on les utilise lors de la conception de circuits. Nous aurons ainsi une vue générale de tous les éléments discutés jusqu'ici.

Exercice 4
Dessinez des tableaux de Karnaugh à trois variables pour simplifier les expressions suivantes :
a) $\bar{A}.B.C + \bar{A}.\bar{B}.C + \bar{B}.\bar{C} + \bar{A}.B.\bar{C}$
b) $A.\bar{B}.\bar{C} + \bar{A}.B.\bar{C} + A.B.\bar{C}$

Codes de commande

On ne peut pas dire que le basic Commodore soit un langage « avancé », mais il est simple et doté d'une logique solide. Il existe cependant des manques importants.

Toutes les machines Commodore BASIC acceptent des noms de variables, mais seuls les deux premiers caractères du nom sont pris en compte par l'interpréteur. Aussi des variables telles que FUJIYAMA et FUTILITÉ ne pourront être distinguées. De la sorte le résultat de :

```
100 FUJIYAMA=17:FUTILITÉ=2*FUJIYAMA
200 PRINT FUJIYAMA,FUTILITÉ
```

sera :

```
34 34
```

Cela s'applique à tous les types de variables : de type virgule flottante (NOMBRE), valeur entière (NOMBRE %), chaîne (NOMBRE) et tableau (NOMBRE (62,47)). Les types de variables sont eux-mêmes conventionnels, mais les variables entières ne sont pas acceptées par le Vic-20 du fait que la machine ne comporte pas d'arithmétique pour les valeurs entières. Ce type de variable ne concerne que les machines Commodore qui les implémentent.

Une conséquence fâcheuse de ce qui précède vient de ce que des noms valides en apparence pourront être en réalité interdits du fait que leurs deux premières lettres commencent comme un mot réservé. Ainsi START revient en tant que nom de variable à ST, et ST est précisément un mot réservé.

Les variables de type tableau peuvent comporter jusqu'à 255 dimensions et ne sont limitées en quantité que par la mémoire vive disponible. Le premier élément d'un tableau est l'élément 0, aussi DIM EX(6) crée-t-il un tableau de sept éléments : EX(0), EX(1), EX(2)... EX(6). L'instruction DIM n'est pas nécessaire ici : lorsque l'interpréteur rencontre une variable tableau à une dimension sans DIM, une dimension par défaut de 10 est prise en compte. Si l'indice d'un tel tableau est supérieur à 10, un message MAUVAIS INDICE s'affiche. Pour être pratique, cela n'encourage pas la programmation méthodique. L'interpréteur doit réorganiser la mémoire à chaque instruction DIM (ou lors de la première référence à un tableau non dimensionné — sans DIM).

Aussi tous les tableaux devraient être dimensionnés (par DIM) en même temps, au commencement du programme, avant d'employer toute variable simple. Rien de grave n'arrivera si cela n'est pas respecté, mais la vitesse d'exécution en sera pourtant affectée.

Du fait du mode de fonctionnement de la plupart des interpréteurs BASIC, l'exécution d'un programme peut être accélérée en initialisant les variables les plus utilisées dans l'ordre de leur importance. Cela se fait par des instructions d'assignation ou par DIM. La ligne :

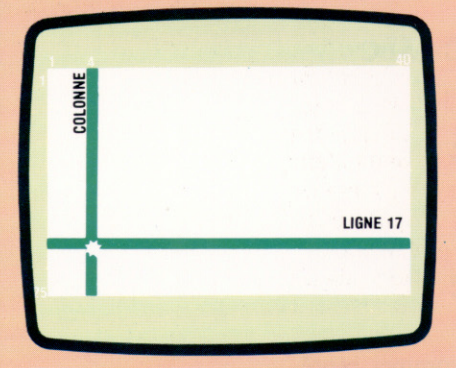
```
10 DIM A$(10,24),K,L,SCORE
```

n'a d'autre effet que de placer les variables K, L et SCORE vers le début de la table, les rendant plus facilement accessibles pour l'interpréteur, et accroissant ainsi la vitesse d'exécution.

Un examen des mots clefs BASIC d'un manuel utilisateur Commodore (parmi les derniers) révèle des différences par rapport au jeu complet Microsoft (manques et ajouts). Le manque le plus important étant INKEY$, et l'ajout le plus significatif, STATUS.

INKEY$, fonction prenant en compte la frappe au clavier, est remplacée par GET. Cette dernière,

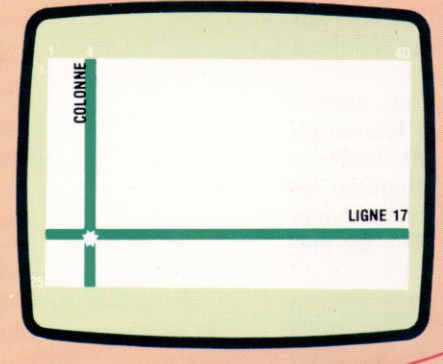
Positionnement du curseur



La faculté d'inclure les commandes du curseur dans la syntaxe de chaînes de caractères rend plus aisée la conception graphique sur Commodore — tout particulièrement en conjonction avec la puissante commande d'éditeur d'écran.

```
100 PRINT RT(17,4)"*"
```

Si le Commodore BASIC comprenait la commande PRINT AT, le positionnement du curseur serait plus simple. Mais du fait qu'il ne le permet pas, nous devons utiliser le curseur programmable.




```
50 POSITION$="//////////////////////////////////////////////////////////////////////////////:"
100 PRINT LEFT$(POSITION$, 17)TAB(4-1)"/"
50 POSITION$="//////////////////////////////////////////////////////////////////////////////"
100 ROW=17:COLUMN=4:GOSUB 1000,PRINT "/"
500 END
1000 PRINT LEFT$(POSITION$,ROW)TAB (COLUMN1);:RETURN
```

Initialisez POSITIONS par HOME et 24 CRSR DOWN. Mettez ensuite dans cette expression les paramètres colonne et ligne.

Si vous avez un travail important de formatage d'écran, vous avez intérêt à inclure la commande de positionnement du curseur dans un sous-programme. Vous initialiserez ensuite les variables ROW et COLUMN par la position à l'écran, avant d'appeler le sous-programme.

Liz Dixon

tout comme INKEY$, transcrit le premier caractère du tampon du clavier en code ASCII. GET est surtout utilisé dans des instructions telles que :

```
150 GET GT$:IF GT$=«» THEN 150
```

qui arrête l'exécution du programme jusqu'à la prochaine frappe d'une touche. GT$ prend alors la valeur du caractère frappé. Cela peut ne pas toujours se produire ainsi dans la mesure où GET balaye le tampon du clavier et non le clavier lui-même. Ainsi, si le tampon contient quelques caractères lors de l'exécution de GET, l'exécution du programme n'attendra pas une frappe au clavier. C'est ce que montre le programme suivant :

```
50 FOR K=1 TO 100:PRINT K:NEXT K
60 PRINT «TAPEZ UNE TOUCHE»
150 GET GT$:IF GT$=«» THEN 150
200 PRINT «VOUS AVEZ TAPÉ »;GT$
```

Si vous exécutez ce programme, vous verrez s'afficher les nombres 1 à 100, suivis du message « ENTRER », jusqu'à ce que vous frappiez une touche. Cependant, si vous frappez une touche pendant l'affichage des nombres, ce caractère ira dans le tampon du clavier. Il y sera trouvé par l'instruction GET. Et il n'y aura pas d'arrêt dans l'exécution du programme. Cela peut être ennuyeux et même désastreux pour les jeux au cours desquels on ne se prive pas d'appuyer sur des touches! La solution est de ré-initialiser le tampon avant exécution de GET, en insérant dans le programme :

```
149 POKE KBPTR,0
```

KBPTR est l'adresse du compteur recensant l'ordre des caractères entrés au clavier (198 pour le Commodore 64).

GET ne génère pas de clignotement du curseur, mais POKE FLASH,0 — où FLASH est l'adresse du drapeau activant le clignotement (204 sur le Commodore 64) — s'en chargera.

TIMES (TI$ en abrégé) est l'horloge du système. A la mise sous tension elle est initialisée à « 000000 », et peut indiquer l'heure en heures, minutes et secondes, jusqu'à « 235959 » — 23 heures 59 minutes et 59 secondes après l'initialisation. Pour l'utilisateur, il s'agit d'une variable susceptible d'être assignée à une heure déterminée telle que TI$=«000000» ou TI$=«084503».

TI est la contrepartie numérique de TI$. Il contient la valeur du temps écoulé depuis l'initialisation, en soixantièmes de seconde. Ses valeurs vont donc de 0 à 5183999 (60*60*60*24-1). TI dépend de TI$ et ne peut être initialisé séparément.

STATUS (ST en abrégé) est une variable définie par le système. Lors de la détection d'une erreur sur un périphérique d'entrée/sortie, ST prendra pour valeur un nombre qui indiquera le type de l'erreur. Ce qui peut s'écrire :

```
330 IF ST > 0 THEN GOSUB 30000: REM I/O
ERREUR DE MANIPULATION D'UN FICHIER
:
30000 REM MESSAGES D'ERREURS
30100 IF ST=16 PRINT «ERREUR FATALE EN LECTURE»
```

CMD est une instruction particulièrement utile. Elle détourne la sortie sur l'écran vers un autre périphérique. Parmi ses applications :

```
OPEN 4,4:CMD 4:LIST
```

Cela a pour effet de lister (LIST) le programme courant sur l'imprimante plutôt que sur l'écran. Une fois le listage effectué, faites :

```
PRINT #4:CLOSE 4
```

CMD peut servir dans un programme à faire une copie d'écran sur imprimante. Si GOSUB 3000 suscite dans votre programme l'affichage d'un message ou de quelques données (PRINT, plutôt que POKE en mémoire de l'écran), pour en faire une copie d'écran, tapez :

```
OPEN,4:CMD 4:GOSUB 3000:PRINT #4:CLOSE 4
```

Bien que le point d'interrogation puisse remplacer PRINT en tant qu'abréviation, vous ne pouvez abréger le mot clef PRINT# en ?# — vous utiliserez «pR».

Les abréviations de mots clefs chez Commodore sont aussi vieilles que la machine, bien que les fans de Sinclair en réclament la paternité pour le Spectrum. Pratiquement tous les mots clefs peuvent être abrégés sous la forme de leur première lettre suivie par l'écriture en capitale de leur deuxième lettre. Lorsque deux mots clés ont les mêmes deux premières lettres, la forme d'abréviation retenue sera leurs deux premières lettres suivies de la troisième en lettre capitale : RESTORE et RETURN par exemple sont abrégés en reS et reT.

L'éditeur de texte et le robuste système d'exploitation qui le sous-tend constituent la caractéristique la plus remarquable des Commodore. Elle existe depuis la création du micro PET et constitue encore aujourd'hui le meilleur éditeur d'écran. Ainsi pour éditer une ligne de programme, appliquez LIST à ladite ligne, positionnez le curseur sur la ligne, éditez le texte et tapez « Retour chariot ». Les emplacements du texte à l'écran et du curseur sur la ligne sont sans importance — le texte sur la ligne du curseur est entré au système comme si vous veniez de le taper.

Une des subtilités de ce système d'édition est la capacité de copie offerte. Si vous avez à saisir ces deux lignes :

```
100 IF INT(NOMBRE/INDEX-TAUX)=5 THEN 3000
200 IF INT(NOMBRE/INDEX-TAUX)=7 THEN 3800
```

vous ne taperez que la première suivie de RC. Puis, en déplaçant le curseur, vous changerez le numéro de ligne de 100 à 200, 5 en 7, 3000 en 3800, et vous taperez RC. La ligne 100 est restée intacte en mémoire, et son texte édité à l'écran est devenu la ligne 200. Ce stratagème peut être répété autant que vous le désirez.

Ce n'est là qu'une des ressources de l'éditeur, mais elle montre sa souplesse et sa rapidité. Ainsi, après avoir connu l'éditeur Commodore, tous les autres éditeurs vous sembleront aussi grossiers qu'un piano dont il vous faudrait jouer avec des gants.

SUPER ESCROC

En 1971, Jerry Schneider a escroqué de 100 000 $ la Los Angeles Pacific Telephone and Telegraph Company. En récupérant dans les poubelles de cette entreprise des manuels et du matériel, et en jouant au reporter, il a pu trouver le code d'accès à l'IBM 360 de la société. Il a ainsi passé de manière discrète des commandes fictives qui lui ont permis de revendre le matériel.

Lignes d'assemblage

Voici le résumé des principales conventions utilisées en rapport avec la mémoire, en particulier : adressage lo-hi, codage des mots clés et importance du contexte.

Lorsque vous faites tourner un programme, la première chose que fait le système d'exploitation (S.E.) est d'inspecter les pointeurs de début de texte BASIC afin de déterminer dans quelle partie de la mémoire réside le programme à exécuter. A cette fin, le S.E. doit stocker les adresses des pointeurs, mais pourquoi ne stocke-t-il pas tout simplement les adresses pointées?

La raison essentielle est la souplesse. Rappelons que le S.E. est un programme permanent, résidant en ROM, et toutes les données qu'il contient sont également permanentes. Supposons qu'il existe différentes versions de l'ordinateur à un moment donné et que, bien qu'il soit commode d'avoir le début du texte BASIC à l'octet 2048 dans la version 1, il devient nécessaire de le déplacer à l'octet 4096 dans la version 2. Cela impliquerait que la seconde machine ne pourrait utiliser le S.E. de la première à cause des localisations différentes de la zone texte BASIC. En outre, il faudrait développer de nouvelles ROM pour chaque variante de la machine, ce qui est coûteux; et le logiciel écrit pour l'une ne serait plus valable pour l'autre. Si toutefois les ROM de S.E. ne contiennent que les adresses des pointeurs, alors les mêmes peuvent servir pour tous les modèles, et seuls les contenus des pointeurs devront être changés d'une version à l'autre. La position des pointeurs eux-mêmes peut rester constante parce que le S.E. n'a besoin que d'une relativement petite zone de mémoire pour stocker ses programmes et ses données (généralement 1 000 octets environ). En fixant la position de cette zone — en général les quatre premières pages de mémoire — et en concevant le système autour de celle-ci, on ne restreint pas considérablement l'ensemble de la conception. Par contre, si l'on fixe la position de la zone texte BASIC (qui peut aller de 3 000 à 40 000 octets), la restriction est sévère.

Pratique standard

Par convention, on stocke les adresses dans les pointeurs sous la forme *lo-hi*. Si par exemple les octets 43 et 44 doivent indiquer l'adresse 7671 (page 29, décalé 247), alors l'octet 43 contiendra 247 (décalé ou *lo*), tandis que l'octet 44 contiendra 29 (page ou *hi*). Ce procédé, confus à première vue, est pratique pour le microprocesseur du fait que l'octet lo de l'adresse est stocké dans l'octet lo du pointeur (de même pour hi).

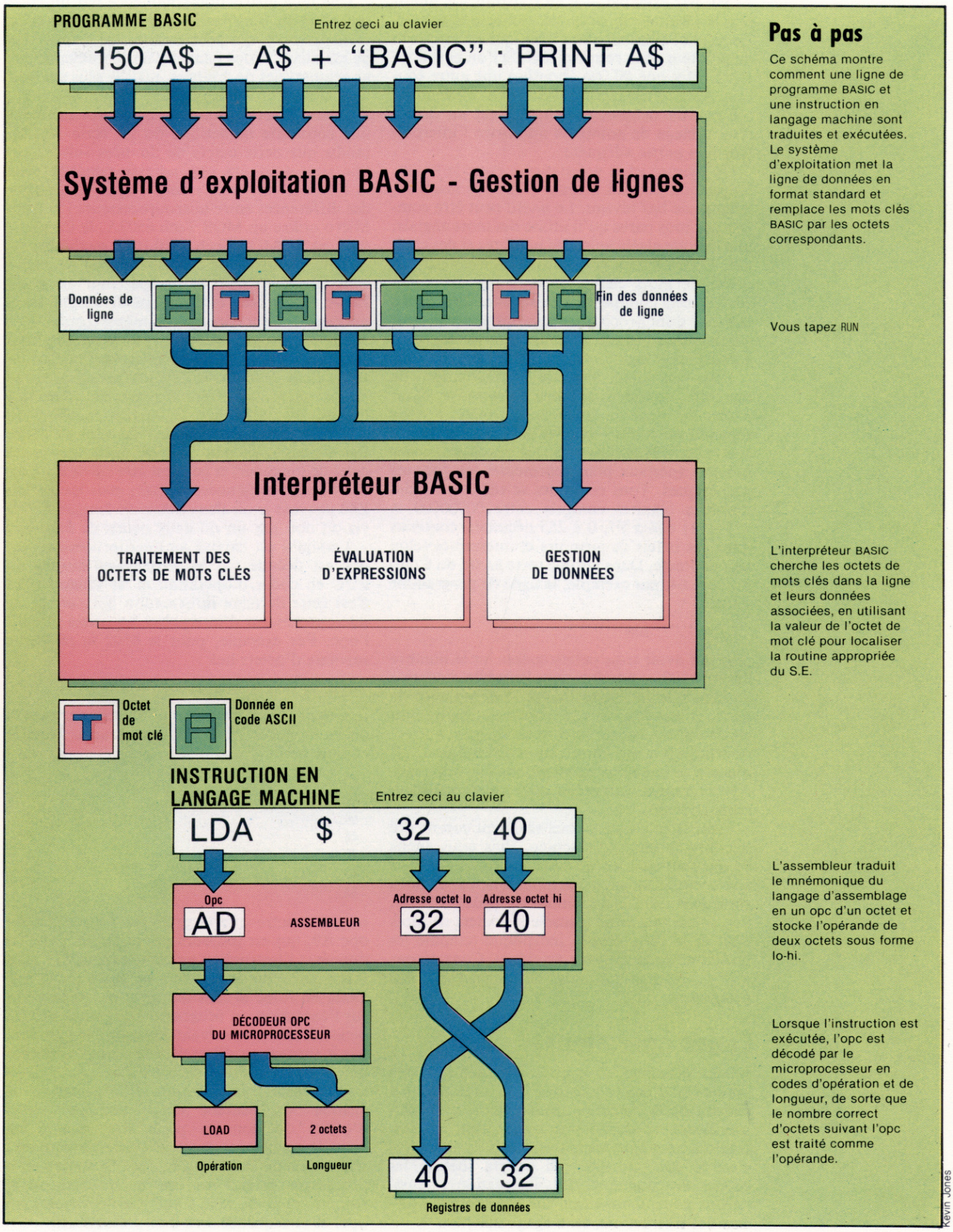
Si nous reprenons l'exemple précédent en utilisant des nombres hexadécimaux au lieu de décimaux, on peut voir le grand avantage du système hexadécimal (désormais, les adresses et autres nombres seront toujours écrits en hex et précédés de «$»). Les octets de pointeur sont $2B et $2C et l'adresse qu'ils indiquent est $1DF7. Ainsi, $2B contient F7 (adresse lo) et $2C contient $1D (adresse hi). Notez que lorsque l'adresse est en hex, les deux chiffres hex de droite sont l'octet lo, et les deux de gauche l'octet hi, ce qui est bien plus logique que d'utiliser les nombres décimaux.

Il faut remarquer que, contrairement à la plupart des ordinateurs, le BBC et le Spectrum stockent les lignes de programme sous la forme hi-lo au lieu de lo-hi. Il est vrai que ce sont des paramètres de programme plutôt que des adresses d'octets. Néanmoins ils sont contraires à la convention usuelle.

Par ailleurs, bien que les pointeurs soient sur deux octets, ils sont souvent appelés par l'octet lo seul. Par exemple, on peut dire que sur le Commodore 64, l'octet 43 indique le début du texte BASIC. On entend par là que les octets 43 et 44 forment ensemble les pointeurs.

On peut encore considérer les octets de mots clés. Leur signification pour les programmeurs en langage machine est double : ils représentent des commandes en anglais comprenant plusieurs caractères (telles que PRINT ou RESTORE) par des codes numériques d'un seul octet; et ils utilisent également des décalés. Une commande BASIC est un mot, mais son exécution n'est pas une seule opération pour le système d'exploitation. L'instruction PRINT, par exemple, exige que les données à imprimer soient trouvées en mémoire ou évaluées, puis elle les envoie caractère par caractère à l'écran en code ASCII. Les différentes tâches sont accomplies par une routine du programme interpréteur BASIC. Lorsque l'interpréteur rencontre l'octet PRINT dans une ligne de programme, il utilise la valeur de cet octet pour localiser, puis exécuter la routine correspondante.

Supposez que vous n'ayez que trois instructions dans votre version de BASIC : INPUT, PRINT et STOP; et que celles-ci correspondent aux octets : $80, $81 et $82. Admettons, en outre, que les routines de l'interpréteur qui exécutent ces instructions commencent respectivement aux octets $D010, $EA97 et $EC00, ces trois adresses étant stockées sous la forme lo-hi dans les six octets de $FA00 à $FA05; cela nous donne un tableau de trois pointeurs à deux octets. Or quand notre interpréteur imaginaire rencontre un octet de mot clé



Pas à pas

Ce schéma montre comment une ligne de programme BASIC et une instruction en langage machine sont traduites et exécutées. Le système d'exploitation met la ligne de données en format standard et remplace les mots clés BASIC par les octets correspondants.

Vous tapez RUN

L'interpréteur BASIC cherche les octets de mots clés dans la ligne et leurs données associées, en utilisant la valeur de l'octet de mot clé pour localiser la routine appropriée du S.E.

L'assembleur traduit le mnémonique du langage d'assemblage en un opc d'un octet et stocke l'opérande de deux octets sous forme lo-hi.

Lorsque l'instruction est exécutée, l'opc est décodé par le microprocesseur en codes d'opération et de longueur, de sorte que le nombre correct d'octets suivant l'opc est traité comme l'opérande.

Kevin Jones

— $81, par exemple — il en soustrait $80, multiplie le résultat par deux, et ajoute cela à $FA00. Le résultat final est alors $FA02, qui est l'octet lo du pointeur de la routine PRINT. Si l'on part d'un autre octet que $81, on aboutit à une autre routine. Ainsi, l'instruction BASIC PRINT est remplacée par un demi-octet, $81, qui est un décalé d'un tableau de pointeurs qui dirige l'interpréteur à la partie adéquate.

On mesure ici la « distance » entre le BASIC, langage dit de haut niveau, et le code machine, langage de bas niveau. Le BASIC peut être compris par tous parce qu'il utilise des mots anglais, une logique algébrique, des nombres et des chaînes. Lorsque nous remplaçons les mots par des octets de mots clés et le reste par des codes ASCII, cela commence à ressembler bien plus à la façon dont procède un microprocesseur — et comme nous l'avons vu pour les mots clés, c'est à peu près le cas.

Pour en finir avec la manipulation de mémoire, considérons l'idée de *contexte*. Nous avons vu dans la zone texte BASIC l'usage répandu de codes — codes ASCII pour représenter les instructions et (sur le Spectrum) codes binaires spéciaux pour représenter les données numériques. Tous ces codes se réduisent à des nombres binaires compris entre 00000000 et 11111111 ($00 à $FF, 0 à 255 décimal), contenus dans des octets de mémoire et interprétés selon leur contexte. Dans la zone texte BASIC du Commodore 64, par exemple, la ligne de programme BASIC :

```
200 rem*******left$*********
```

pourrait avoir trois octets contenant le nombre décimal 200 — une fois dans l'octet lo d'adresse de lien, une fois dans l'octet lo de numéro de ligne, et une fois dans la représentation du mot clé de «left$». Chaque octet ressemble aux autres; pourtant, il a une signification différente. C'est d'après ce que nous en attendons que nous saurons comment interpréter cette valeur en différents endroits.

C'est là que nous avons vraiment commencé le cours de langage machine. Puis nous avons vu que tout était stocké dans un ordinateur dans une sorte de langage machine. Une partie nous était déjà familière (comme les codes ASCII), une partie inconnue (comme les octets de mots clés) et le reste encore inexpliqué (comme les programmes en langage machine). Considérons à présent les programmes en langage machine proprement dits.

Codes opérations

Les programmes en langage machine sont des séquences d'octets localisés en n'importe quel endroit de la mémoire; instructions au microprocesseur et données à traiter sont mêlées. Comme pour tous les autres octets de mémoire, c'est le contexte seul qui pourra séparer les octets de données des instructions; aussi devons-nous commencer par considérer le format des instructions en langage machine.

Une telle instruction commence par un code qui identifie l'opération à effectuer. C'est le *code-op* (*opc*), qui peut avoir un ou deux octets de longueur. L'opc peut être une commande qui ne requiert pas de données, mais le plus souvent il est suivi d'un ou deux octets de données. S'il y a un seul octet de données, c'est probablement une constante numérique ou un code ASCII, tandis que deux octets de données suivant un opc sont toujours une adresse (stockée sous la forme lo-hi). A ce stade, il nous faut parler des différences entre microprocesseurs : le BBC Micro utilise un MOS Tech 6502A, le Commodore 64 un MOS Tech 6510 (très semblable au 6502A, de sorte que désormais nous ne parlerons que du 6502), et le Spectrum un Zilog Z80A. MOS Tech et Zilog ont développé leurs microprocesseurs presque en même temps — au début des années soixante-dix — après que Intel eut fourni son premier microprocesseur en 1971. Si le 6502 et le Z80 partagent la même philosophie, ils diffèrent nettement dans les détails. En particulier, le langage machine du Z80 est entièrement différent de celui du 6502. Par exemple, les opc du 6502 ont toujours un octet de longueur, et peuvent être suivis d'un ou deux octets de données ou pas; mais les opc du Z80 peuvent avoir deux octets de longueur, suivis ou non par un ou deux octets de données.

Lorsqu'il est envoyé au microprocesseur, un opc est décodé par le programme interne de l'UC en codes d'opération et de longueur, et c'est cette dernière information qui permet au microprocesseur d'interpréter les octets suivant l'opc. Par exemple, pour le 6502 la séquence suivante d'octets hex :

```
A9 0E 8D 01 4E 60 44 52 41 54
```

représente trois instructions, suivies par quatre octets de codes ASCII. On pourrait le réécrire comme ceci :

```
A9 0E
8D 01 4E
60
44
52
41
54
```

ce qui montre que la première instruction est opc A9, qui est toujours suivie par un octet de données; l'instruction suivante est opc 8D, toujours suivie de deux octets de données; la suivante est l'opc 60 sans données, qui se branche sur l'exécution du programme, de sorte que les octets de données suivants ne sont pas examinés par le processeur. Si ce dernier reçoit le premier octet, A9, lorsqu'il attendait un opc, tout fonctionne ensuite sans accroc. L'information de chaque opc assurera que le nombre correct d'octets de données pour chaque opc est recueilli par le processeur, et l'octet suivant sera traité comme le prochain opc. Si toutefois le processeur attend un opc et qu'il reçoit le second octet, 0E, il le traitera alors comme un opc, et la séquence sera interprétée ainsi :

```
0E 8D 01
4E 60 44
52
```

ce qui signifie : l'opc 0E suivi de deux octets de données; puis l'opc 4E avec également deux octets de données; puis l'opc 52, qui n'est pas un opc légal, ce qui causera l'équivalent d'une erreur de syntaxe dans le processeur. Cela démontre comment un malentendu initial engendre une série de graves erreurs logiques dans l'exécution du programme.

Cela met aussi en évidence quelques autres détails importants sur le langage machine : il est vraiment peu agréable à l'utilisateur (du moins au commencement), en ce qu'il est difficile à lire et à écrire; il est tout à fait séquentiel, seul l'ordre différencie une instruction de la suivante; il est littéral comme seule peut l'être une machine, obéissant aussi bien aux instructions fausses qu'aux correctes, et ne rejette que les erreurs de syntaxe.

Certains de ces désagréments peuvent être évités en écrivant des mnémoniques alphabétiques au lieu des opc numériques dans le programme, et en n'ayant recours aux opc que lorsque le programme est chargé en mémoire. Ces mnémoniques constituent le langage d'assemblage du processeur, et leur traduction en opc s'appelle l'assemblage. Remarquez qu'il y a une correspondance bi-univoque entre les mnémoniques du langage d'assemblage et les opc; bien que le langage d'assemblage soit de plus haut niveau que le langage machine, la différence est minime.

Si nous récrivons le fragment de langage machine ci-dessus en langage d'assemblage 6502, nous obtenons ceci :

```
0000 A9 0E    LDA#$0E
0002 8D 01 4E STA $4E01
0005 60       RTS
```

La même séquence d'opérations en langage d'assemblage Z80 donne ceci :

```
0000 3E 0E    LD A,$0E
0002 32 01 4E LD($4E01),A
0005 C9       RET
```

La première colonne montre l'adresse hex en mémoire du premier octet de la ligne — opc A9 dans la liste de 6502, par exemple, est dans l'octet 0; l'octet de page 4E dans les deux listes est dans l'octet 4, etc. La colonne suivante peut contenir un, deux ou trois octets, et montre le listage en code machine. La troisième colonne commence par un mnémonique et donne la version en langage d'assemblage. N'essayez pas de déchiffrer cela dès maintenant, il suffit que vous ayez vu un listage en langage d'assemblage et puissiez observer les différences et similitudes entre les versions Z80 et 6502. Vous remarquerez aussi que l'adresse dans la deuxième ligne apparaît dans sa forme conventionnelle lo-hi en langage machine, mais dans la forme « normale » hi-lo en langage d'assemblage. Nous examinerons les opc en détail prochainement.

Convertisseur hexadécimal

Pour convertir le programme Mempeek vu précédemment afin que le contenu des octets soit exprimé en hex plutôt qu'en décimal, faites les modifications suivantes :

BBC Micro

Ajoutez :

```
3000 DEF PROCHXPRINT(DECNUM)
3100 LOCAL X$
3200 X$ = "0123456789ABCDEF"
3300 HB = INT(DECNUM/16):LB = DECNUM - HB*16
3400 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1 +" "
3500 PRINT B$;
3600 ENDPROC
```

et remplacez la ligne 600 par :

```
600 PROCHXPRINT(PK%)
```

Spectrum

Ajoutez :

```
10 LET X$ = "0123456789ABCDEF"
3000 REM******S - P OCTET HEX******
3100 LET HB = INT(PK/16):LET LB = PK - HB*16
3200 LET B$ = X$(HB + 1) + X$(LB + 1) +" "
3300 PRINT B$;
3400 RETURN
```

et remplacez la ligne 600 par :

```
600 GOSUB 3000
```

Commodore 64

Ajoutez :

```
10 LET X$ = "0123456789ABCDEF"
3000 REM ****** S - P OCTET HEX ******
3100 HB = INT(PK/16):LB = PK - HB*16
3200 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1) +" "
3300 PRINT B$;
3400 RETURN
```

et remplacez la ligne 600 par :

```
600 GOSUB 3000
```

Ces modifications auront pour effet que le contenu de la mémoire sera exprimé en hex. L'adresse de départ et le nombre d'octets devront toujours être entrés en décimal.

Le futur, c'est l'avenir

Xerox, la grande firme de matériel reprographique, fut l'une des premières à s'aventurer dans le domaine de la bureautique. Son excellente réputation est un atout majeur.

Au début des années soixante-dix, Xerox mit sur pied un vaste programme de recherche visant à réaliser un vieux rêve : disposer de l'information à volonté, comme l'électricité ou l'eau courante. Elle rassembla une équipe de chercheurs auxquels elle laissa carte blanche, et prit soin de les installer à Palo Alto, en Californie, c'est-à-dire le plus loin possible de ses propres quartiers généraux de Rochester, dans le New Hampshire.

L'idée se révéla fructueuse. Le PARC (Palo Alto Research Center) se trouvait dans le comté de Santa Clara, non loin de l'université Standford, dont le département d'informatique était à l'avant-garde des recherches sur l'intelligence artificielle. La nouvelle communauté attira bientôt quelques-uns des meilleurs cerveaux de l'endroit et de nombreux étudiants de valeur passèrent sans effort de la pure théorie à la recherche industrielle. Le PARC devint même le centre de la nouvelle culture informatique; on y parlait un jargon accessible aux seuls initiés. Plusieurs produits de Xerox en cours de développement se virent dotés de noms de code facétieux : la série de micro-ordinateurs 820 fut ainsi appelée *Worm,* « le ver », parce qu'il devait « croquer la Pomme ».

L'équipe de recherche s'efforçait avant tout de mettre au point un « réseau » local. L'idée est aujourd'hui familière, mais les premières expériences de Xerox en ce domaine (à Hawaii à la fin des années soixante) avaient pour l'époque quelque chose de révolutionnaire. Connecter un gros système et un terminal était très onéreux, en raison du coût du câblage permettant une communication très rapide; et des problèmes se posaient dès que la longueur des câbles dépassait vingt mètres. Il était bien sûr possible de se servir du réseau téléphonique, mais cela restreignait l'échange de données à 9 600 bauds.

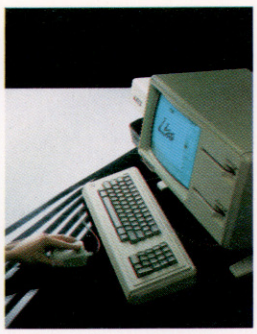
L'objectif du PARC était un réseau qui relierait entre eux des petits ordinateurs; il aurait une vitesse d'accès raisonnable, et l'utilisateur disposerait ainsi d'une plus grande puissance de calcul, tout en pouvant se raccorder à des appareils plus gros, à des mémoires sur disques durs, à des traceurs ou des imprimantes, autant de dispositifs coûteux. Ce fut l'idée de base derrière le réseau local Ethernet.

Les connexions y étaient réalisées grâce à un câble coaxial conventionnel, qui peut transférer 10 millions de bits par seconde, et se montre bien adapté à la transmission de sons numérisés, de graphismes et de données. De plus, le système pouvait s'étendre sur cinq cents mètres sans avoir besoin de relais amplificateurs. Un simple branchement permettait de se raccorder au réseau, ce qui lui donnait une souplesse exceptionnelle.

Le réseau lui-même, sous sa forme tangible, reste passif : toutes sortes de données y circulent, et un émetteur-récepteur installé sur chaque appareil détermine si telle ou telle information lui est bien destinée. Si c'est le cas, il décode le message et lui donne une forme acceptable pour l'appareil — qui peut être un micro-ordinateur, une imprimante, un traceur ou tout ce qu'on désire.

Dès le milieu des années soixante-dix, Ethernet était opérationnel. Xerox pensa que l'appui d'autres grandes firmes permettrait au système de devenir un standard dans le domaine de la communication informatique. IBM, consulté, déclina une offre de participation. Digital Equipment accepta avec enthousiasme, et, en 1975, Intel se joignit à l'entreprise : soutien important, la firme se chargeant de mettre au point le microprocesseur qui équiperait l'émetteur-récepteur.

Ethernet fut d'abord testé en Suède, dans un établissement industriel expérimental : il passa brillamment l'épreuve, et fut adopté par bien d'autres constructeurs. Il est aujourd'hui un standard international reconnu et Hewlett-Packard, ICL en Grande-Bretagne, Siemens en Allemagne et Olivetti en Italie ont décidé de s'y rallier. Xerox a eu l'idée audacieuse d'en vendre les brevets pour 1 000 dollars seulement, afin d'imposer plus sûrement le système.



Avant LISA
Une des plus grandes réussites du PARC fut le STAR, un système de programmation qui recourait au langage SMALLTALK et qui opérait, en combinant programmes et données, au sein d'un même fichier. Le LISA d'Apple doit beaucoup à ce type d'étude.
(Cl. Chris Stevens.)

PROGRAMME N° 16

QUELQUES PETITS PROGRAMMES SCOLAIRES EN MATHÉMATIQUES

Nous commençons une série de programmes relatifs à des problèmes de mathématiques. Aujourd'hui, à travers nos connaissances de BASIC, composons un programme à partir de l'algorithme d'Euclide pour déterminer le P.G.C.D. (plus grand commun diviseur) de deux nombres entiers. Rappelons l'algorithme d'Euclide.

A partir de 2 nombres X et Y dont on veut calculer le P.G.C.D., déterminons leur différence R, le reste de la division de X par Y ou de Y par X, peu importe.

1) On a donc :

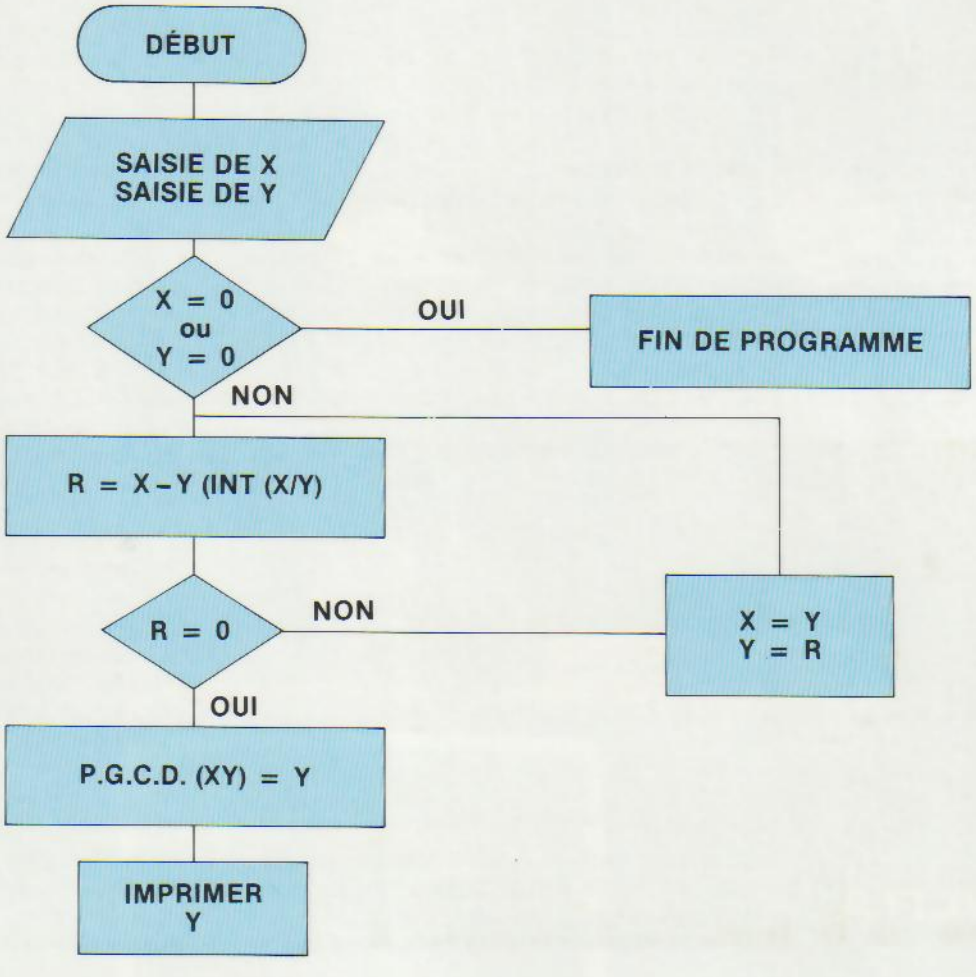
R = X — Y (partie entière (X/Y)).

2) Testons la valeur de R :

Si R = 0, alors X est divisible par Y et le P.G.C.D. de (X, Y) est Y;
si R ≠ 0, alors prenons :

X = Y et Y = R, et recommençons à l'étape 1.

L'organigramme devient :



```
5   HOME
10   HTAB 18: INVERSE : PRINT "P.G
     .C.D.": NORMAL
15   PRINT : PRINT : PRINT : PRINT
      "PLUS GRAND DENOMINATEUR COM
     MUN SELON"
20   PRINT : PRINT "L'ALGORITHME D
     'EUCLIDE"
25   REM SAISIE DES 2 NOMBRES
30   PRINT : PRINT : PRINT "TAPEZ
      LES 2 ENTIERS (0,0 POUR STOP
     )"
35   PRINT : INPUT "X=";X
40   PRINT : INPUT "Y=";Y
45   REM ON STOPPE LE PROGRAMME
50   IF X = 0 OR Y = 0 THEN 150
55   REM ON CALCULE LE PGCD GRACE
       A L'ALGORITHME D'EUCLIDE
60   REM ON TRAVAILLE EN VALEUR A
       BSOLUE
65 X = ABS (X)
70 Y = ABS (Y)
80   REM ON CALCULE LE RESTE
85 R = X - Y * ( INT  (X / Y))
90   REM ON TESTE LA VALEUR DE R
95   IF R = 0 THEN GOTO 130
100 X = Y
105 Y = R
110   GOTO 85
130   PRINT : PRINT : PRINT "P.G.C.
     .D.=";Y
135   PRINT : PRINT : PRINT
140   REM ON COMMENCE
145   GOTO 25
150   PRINT : PRINT : PRINT "AU RE
     VOIR FIN DE TRAITEMENT"
155   END
```

```
]RUN
                    P.G.C.D.

PLUS GRAND DENOMINATEUR COMMUN SELON
L'ALGORITHME D'EUCLIDE

TAPEZ LES 2 ENTIERS (0,0 POUR STOP)

X=124
Y=144

P.G.C.D.=4

TAPEZ LES 2 ENTIERS (0,0 POUR STOP)

X=12
Y=24

P.G.C.D.=12

TAPEZ LES 2 ENTIERS (0,0 POUR STOP)

X=50
Y=455

P.G.C.D.=5

TAPEZ LES 2 ENTIERS (0,0 POUR STOP)

X=0
Y=45

AU REVOIR FIN DE TRAITEMENT
```

Remarquons l'utilisation de l'instruction ABS pour obtenir la valeur absolue de X et Y (car on ne sait pas trouver le P.G.C.D. de nombres négatifs) et l'utilisation de la saisie de X = 0 ou Y = 0 pour stopper le programme.
En effet, n'oublions pas que la division par zéro est interdite.